Volume 3, Issue 7, July 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

# Comparative Analysis of Low Rate Denial of Service Attack in MANETs

**Arvind Sharma, Dr. Neeraj Kumar**
CSE Dept, Thapar University, Patiala
Punjab, India

***Abstract: Mobile MANETs (MANET) is example of wireless mobile communication. 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. 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 paper. 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 and mitigate the effect of Low Rate DoS. Also, analyse the result these variants during attack. The solution provides rapid detection and mitigation procedure during attack.***

***Keywords: MANETs, Dos, TCP, RED, RRED, SACK.***

## I. Introduction

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 [1]. These networks are highly vulnerable to various security threats. One of the major threats to security of MANETs is Dos attacks.

In Dos attacks, 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. Examples of DoS attacks include TCP SYN attacks that consume protocol data structures on the server operating system; ICMP directed broadcasts that direct a broadcast address to send a flood of ICMP replies to a target host thereby overwhelming it; and DNS flood attacks that use specific weaknesses in DNS protocols to generate high volumes of traffic directed at a targeted victim. In this paper, we study Low Rate DoS attacks that attempt to hinder bandwidth to TCP flows while sending at sufficiently low average rate to elude detection by counter-DoS mechanisms [2]. Also known as "Shrew attacks". In this type of attack, an attacker chooses periodic on-off "square-wave" shrew attacks that consist of short, maliciously-chosen-duration bursts that repeat with a fixed, maliciously chosen, slow-timescale frequency. However, this may causes the degradation in throughput of TCP flow by affecting retransmission time out (RTO) timer of TCP congestion control mechanism. A next section detailed descries Low Rate Dos attack; we use ns-2 simulations to explore the impact of aggregation and heterogeneity on the effectiveness of the shrew attack. We show that even under collective flows with varied burst length, TCP variant SACK and different buffer management schemes (drop tail, RED, etc.) are discussed. Finally, we explore potential solutions to low rate DoS attacks. A router based scheme is used to detect and mitigate Low Rate Dos attacks. In this scheme buffer management scheme RRED is used for controlling congestion at edge node of the network.

Also, we use TCP variant SACK for congestion control at sender side for detection and mitigation purpose.

## II. 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 [2]. 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 [3]. 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".

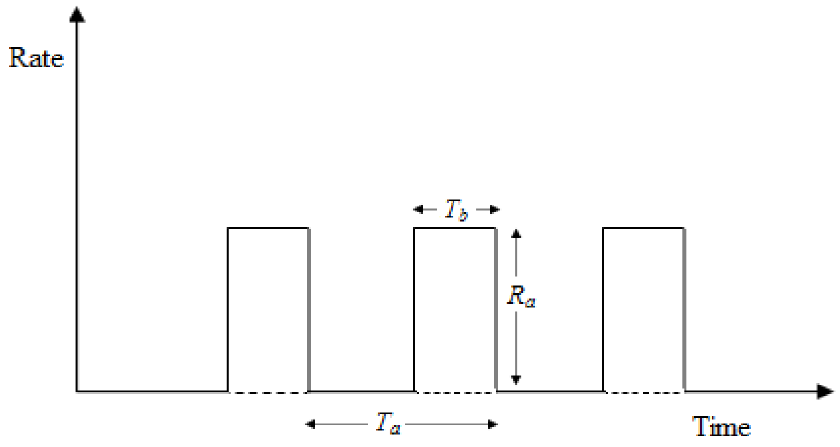

Figure **1:** Low Rate DoS Attack

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.

## III. 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 [4]:

### *A) 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.

### *B) 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. Figure 8 shows the TCP-Reno congestion control mechanism.

***C) TCP New-Reno***

The issues present in TCP-Reno are being overcome in TCP New-Reno by modifying the fast recovery mechanism [5]. 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.

***D) 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:

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

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

***E) 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 defined in [6]. 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 [4]. 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.

## IV. Active Queue Management

In the past decades, quite a few Active Queue Management (AQM) algorithms such as Random Early Detection (RED) [7] and its variants have been proposed to handle congestion and to improve the TCP performance [4]. Although these AQM algorithms are highly robust to diverse network conditions, most of them were designed without considering their robustness against network attacks, such as (DoS) attacks that have been identified as a major threat to today's network services.

***A) 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.

***B) 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 [7].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 [7].

$$q_{avg} = (1 \,||\, w_q)q_{avg} + w_q * q_{instantaneous}$$

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.

***1) Adaptive RED (ARED)***

ARED algorithm infers whether to make RED more or less aggressive based on the observation of the average queue length [8]. 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) 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 [9]. 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.

## V. 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 [2] and [10] to tackle the low rate attacks.

***A) RTO Randomization***

Randomizing the minimum value of RTO value for TCP is the solution proposed in [2]. 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.

***B) Node based Solution***

The second approach is the node-based scheme which uses adaptive queue management schemes to tackle the attack [2, 10]. 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. 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.

## VI. 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.

**A) Network Simulators (Ns-2)**

This section gives a brief introduction to the Ns-2 simulator [11]. 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 [11]. 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 paper, 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.

***B). Implementation***

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

***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.

***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 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 |

***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.

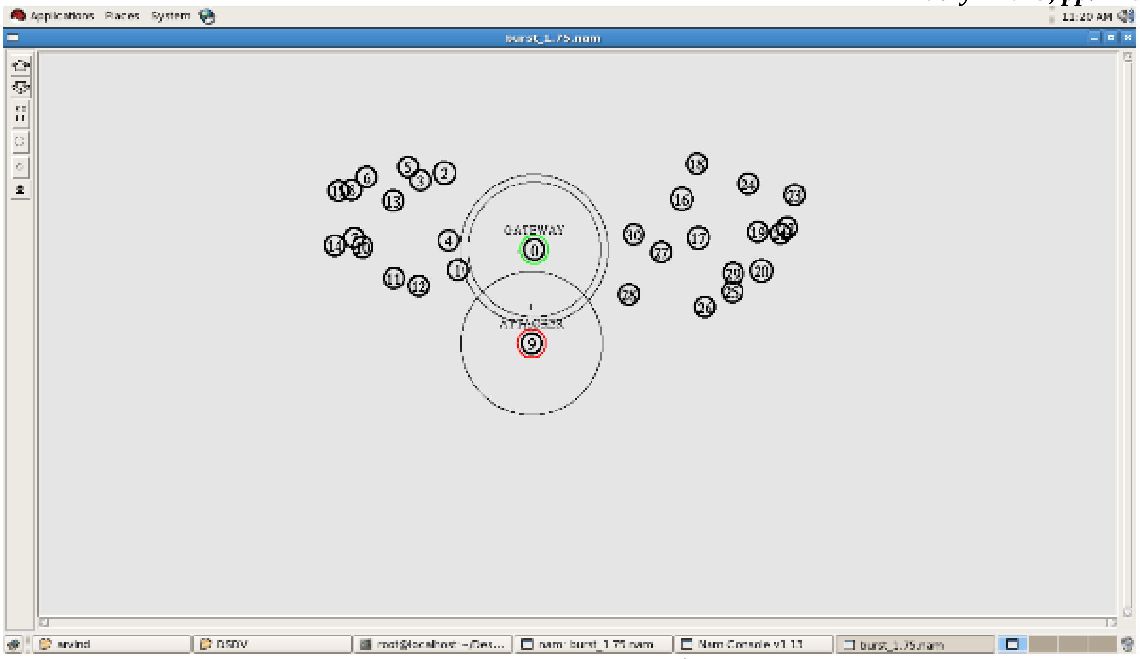

Figure 2: Low Rate TCP Flow from Attacker

***4) Attack Modelling***

In this paper, 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 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.

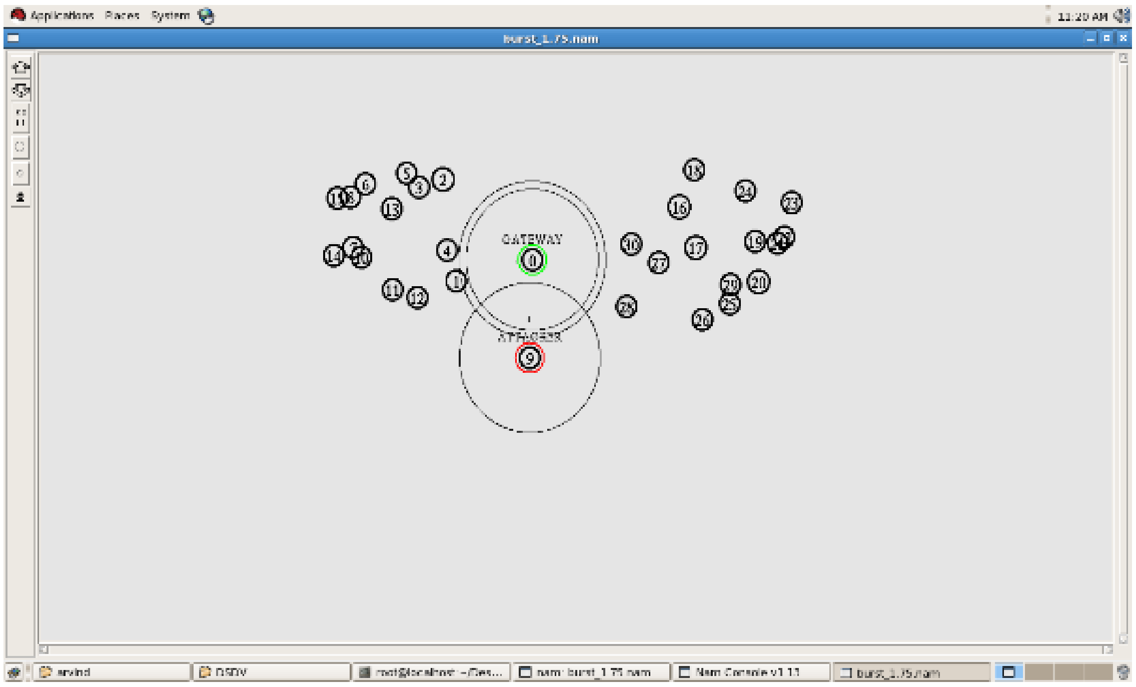

Figure 3: Low Rate TCP Flow from Attacker

## VII. Performance Evaluation

***A) 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.

***1) 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.

***2) 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.

***3) Routing Overhead***

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

**B) Results and Performance Analysis**

Figure 4, shows throughput comparison between DSDV protocol with Drop tail and DSDV with RRED at different burst length of attack. At burst period 1.75 RRED gives better performance as compare to Drop tail AQM.

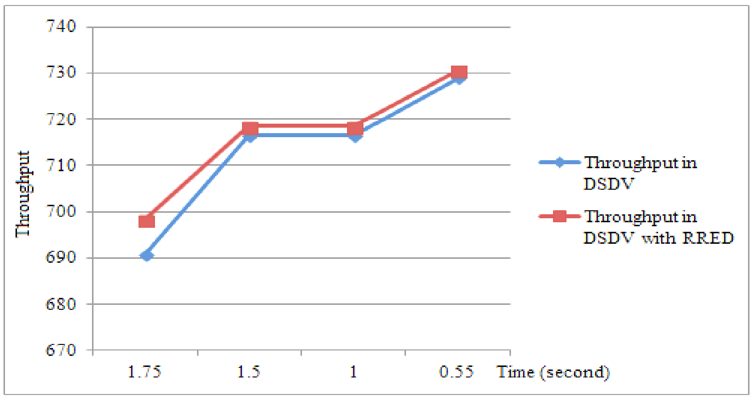

Figure 4: Throughput comparison between Drop tail and RRED

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 shows performance of DSDV protocol with SACK and RRED variant to DSDV protocol with Drop tail variants. As the result seen DSDV with SACK and RRED combination gives better performance at different burst length of attack.

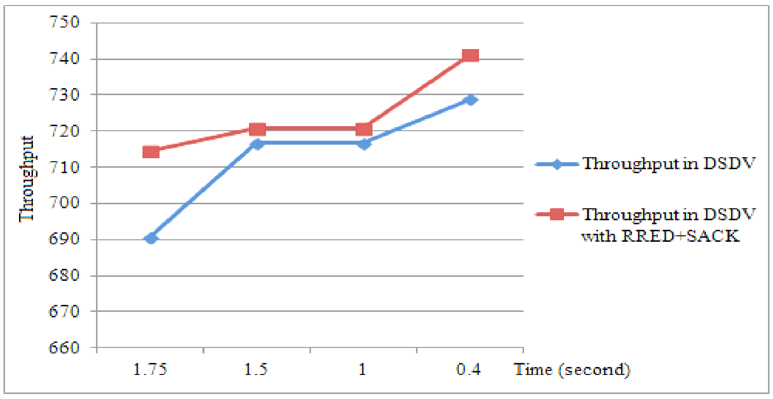

Figure 5: Throughput between Drop tail and RRED with SACK

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 6 shows the comparative delay in DSDV with RRED with DSDV with Drop tail AQM. 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.

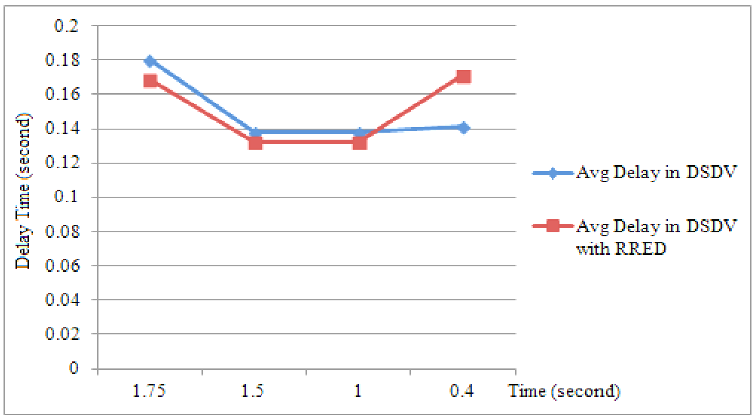

Figure 6: Delay between Drop tail and RRED

Figure 7, delay in DSDV with RRED with SACK is slightly more as compare to Drop tail DSDV because RRED with SACK suffer to deliver packets. When network is congested due to the overflow of network traffic, control packets are injected in the network to inform sender about congestion. RRED and SACK both uses control packets for sending network status to the end point. These control packets also responsible for controlling transmission speed of packets that are being generated by source node. These messages are source quench messages that are uses by intermediate node to regulate traffic in the network. So, network bandwidth may be affected by these control messages. However, we deal with TCP traffic that is reliable communication between source to destination, so we can compromise with this extra overhead in place of packet drop.

Routing overhead of transmitted packets from sender to receiver as shown in figure 8, which gives description about routing overhead in case of DSDV with drop tail and DSDV protocol with RRED variant. As the burst length of attack decreases overhead come almost same in both cases. There is slight deviation at bust length of 1.5 where Drop tail performs better as compare to RRED.

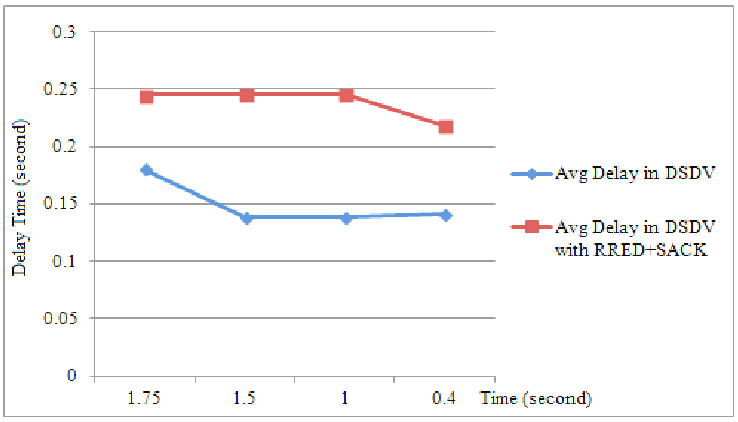

Figure 7: Delay between Drop tail and RRED with SACK.

Figure 9 describes DSDV with Drop tail performs better over DSDV with RRED and SACK combination. Due to control packets as discussed above, those have been generated in RRED and SACK. Simple RRED have performed better then it combines with SACK.

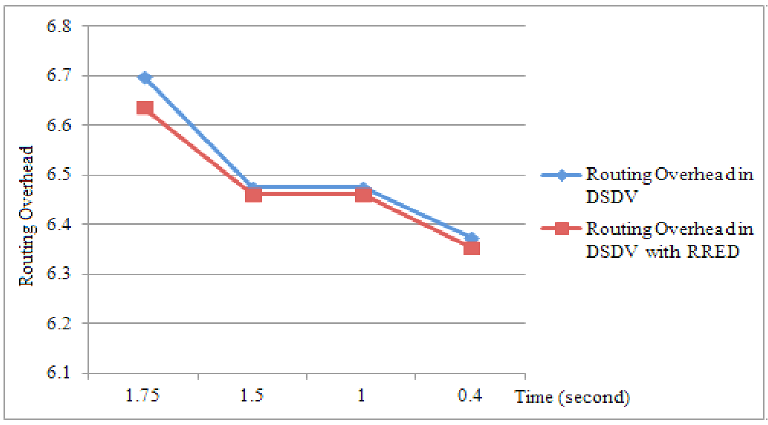

Figure 8: Routing Overhead between Drop tail and RRED

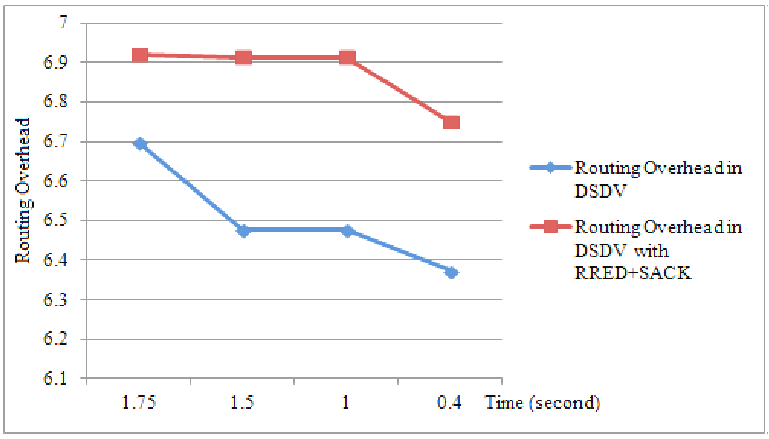

Figure 9: Routing Overhead between Drop tail and RRED with SACK

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

Table 2: Performance Comparison 1

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

Table 3: Performance comparison 2

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

## VIII. Conclusion and Future Scope

In This paper, 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.

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. However, this attack extends with distributed DoS in which more than one attacker try to bottleneck bandwidth with low rate network traffic and try to mitigate the effect of attack by using various AQM techniques in wireless environment.

## References

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

[2] 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.

[3] 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.

[4] X. Luo, R. Chang, 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.

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

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

[7] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, 1993.

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

[9] 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.

[10] G. Yang,M. Gerla, andM. Y. Sanadidi, “Defense against low rate 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.

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