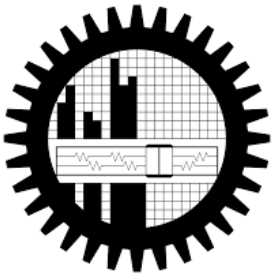
Bangladesh University of Engineering and Technology



CSE-322(Networking Sessional)

Robust Random Early Detection

Project Report

Submitted to :

Name : Md. Tarek Mahmud

Lecturer

CSE, BUET

Submitted by :

Name : Kazi Wasif Amin Shammo

Id : 1705079

CSE, BUET

Date of submission : 23th of February 2022

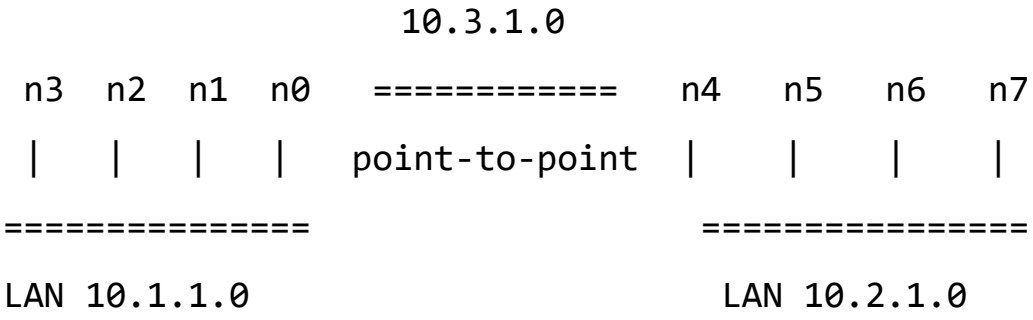
Introduction

The existing Random Early Detection (RED) algorithm and its variants are found vulnerable to emerging attacks, especially the Low-rate Denial-of-Service (LDoS) attacks. **In this report we demonstrate improvement of the TCP throughput against LDoS attacks using RRED technology through NS3 simulation**. The basic idea behind the RRED is to detect and filter out attack packets before a normal RED algorithm is applied to incoming flows. We conduct a set of simulations to evaluate the performance of the proposed RRED algorithm. The results show that, compared to existing RED-like algorithms, the RRED algorithm nearly fully preserves the TCP throughput in the presence of LDoS attacks.

Network topologies under simulation

Wired network

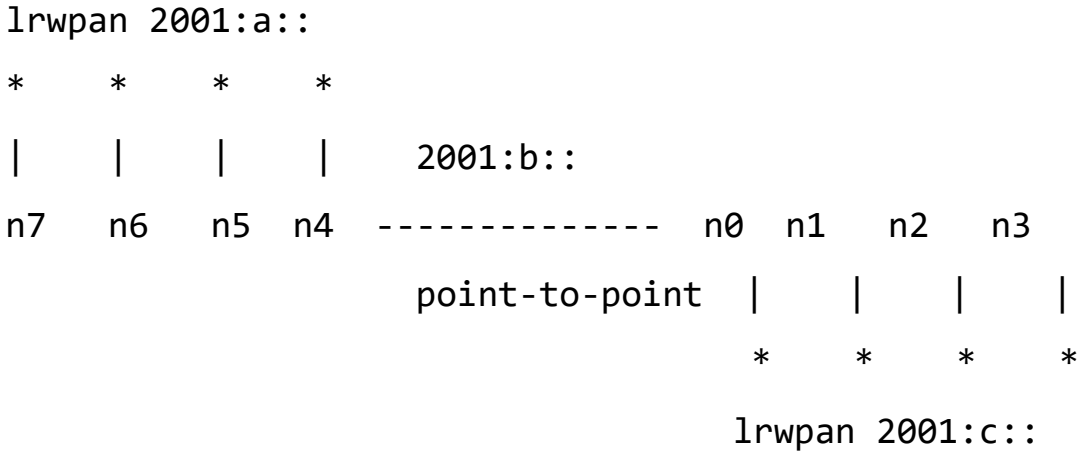
To simulate the wired network, point to point dumble topology, a special topology built in NS3 was used. Point to point dumble topology looks like below.



Here the number of nodes is 8. The same topology was used but the number of nodes varied from 10, 20, 30, 40 to 50.

Lrwpan network

To simulate the lrwpan network also a dumble topology was used. A dumble topology is basically a point-to-point connection with to wpan network on in left and one in right. So, the final topology looks like below.



c Here the number of nodes is 8. The same topology was used but the number of nodes varied from 10, 20, 30, 40 to 50.

Parameters under variation

For all of the simulations we varied the parameters called nLeftLeafCount,nRightLeafCount,maxPackets, maxTh, minTh, pktSize queueDiscLimitPackets, appDataRate, queueDiscType ,port , bottleNeckLinkBw ,bottleNeckLinkDelay

Overview of the proposed algorithm

Basic Idea

All incoming TCP packets to a router belong to different flows. Here, a flow is defined by a 5-tuple (Source IP, Source Port, Destination IP, Destination Port, Protocol). We use an indicator **f.I** to judge whether flow **f** is an LDoS attack flow or a normal TCP flow. Specifically, **f.I** is calculated as follows. If a packet from flow **f** is considered to be an attacking packet , **f.I** is decreased by one; if it is considered to be a normal packet, **f.I** is increased by one. Then an incoming packet from a flow with a negative **f.I** is filtered out. Packets from a flow with a positive or zero **f.I** will further feed to the RED block.

An incoming packet from flow **f** is suspected to be an attacking packet if it arrives within a short-range after a packet from **f** that is dropped by the detection and filter block or after a packet from any flow that is dropped by the RED block. The following process is used to define this short-range. For every flow **f** (either a normal TCP flow or an LDoS flow), let **f.T1** be the arrival time of the last packet from **f** that is dropped by the detection and filter block. Let **T2** be the arrival time of the last packet from any flow that is dropped by the RED block. The short-range is defined as $[Tmax, Tmax + T*]$, in which $Tmax = MAX(f.T1, T2)$. If the arrival time of an incoming packet from flow **f** falls into this range, the packet is suspected to be an attacking packet. T* is empirically chosen to be 10ms.

The pseudo code is given below for better understanding.

Pseudo code

```
RRED-ENQUE(pkt)
    f ← GetFLOW(pkt)
    Tmax ← MAX(Flow[f].T1, T2)
    if pkt.arrivaltime ∈ [Tmax, Tmax + T *] then
        decrease local indicator by 1 of f
    else
        increase local indicator by 1 of f
    end if
    Flow[f].I ← maximum of local I of f
    if Flow[f].I >= 0 then 1
        RED-ENQUE(pkt) //pass pkt to the RED block
        if RED drops pkt then
            T2 ← pkt.arrivaltime
        end if
    else
        Flow[f].T1 ← pkt.arrivaltime
        drop(pkt)
    end if
return
```

Modifications made in the simulator

1.New added classes

Two new classes were added called Flow and FlowSimulator and they are stored in flow-simulator.cc file.

Flow class defines five tuple of the flow along with new parameters T1 and indicator.

FlowSimulator class basically manipulates the Flow class object. It creates Flow object, increments and decrements indicator, modifies T1 and so on.

2.New added variables and functions in red-queue-disc.h

A number of new variables were added in the header file called T2, Tmax, T* and flowSimulator object.

A new member function was added too called RREDCheck(Ptr<QueueDiscItem> , Time&)

3.New added function implementation in red-queue-disc.cc

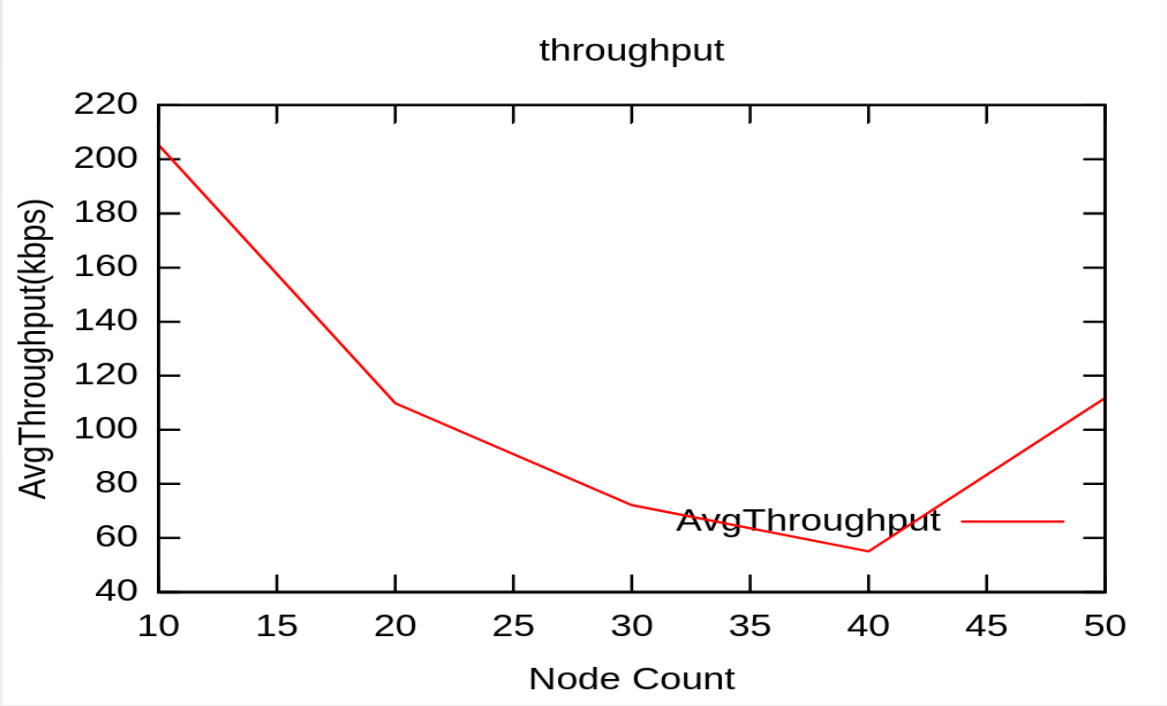
The newly added function RREDCheck(Ptr<QueueDiscItem> , Time&) was implemented according to the algorithm.

4.Network topology for testing

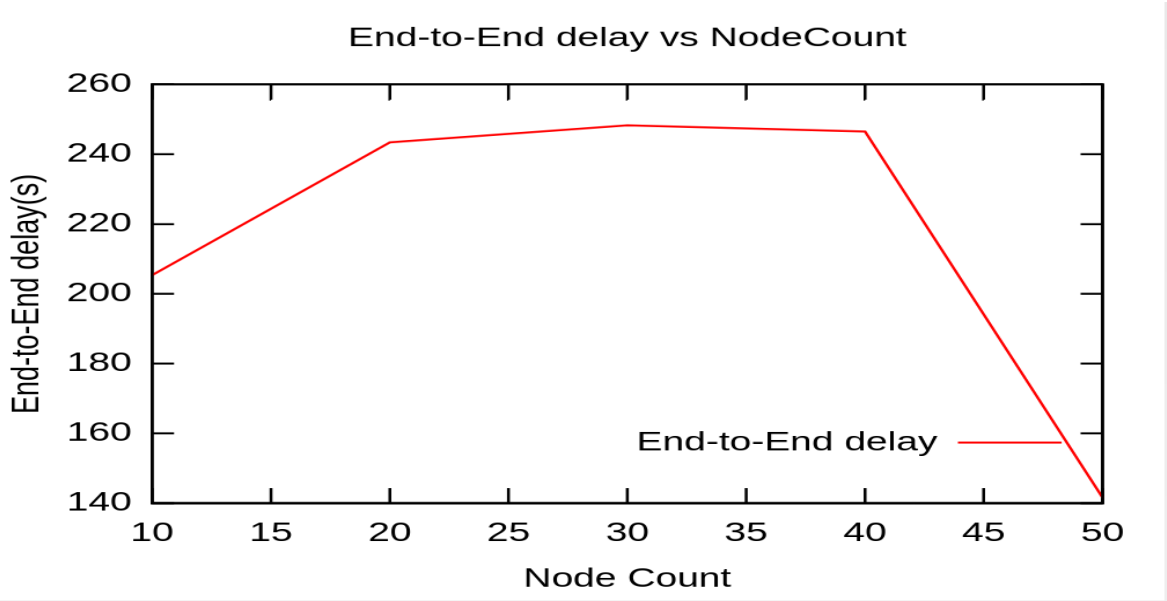
A network topology was designed to test whether the modifications made were working or not.

Graphs for Task A
(Wired, before modification)

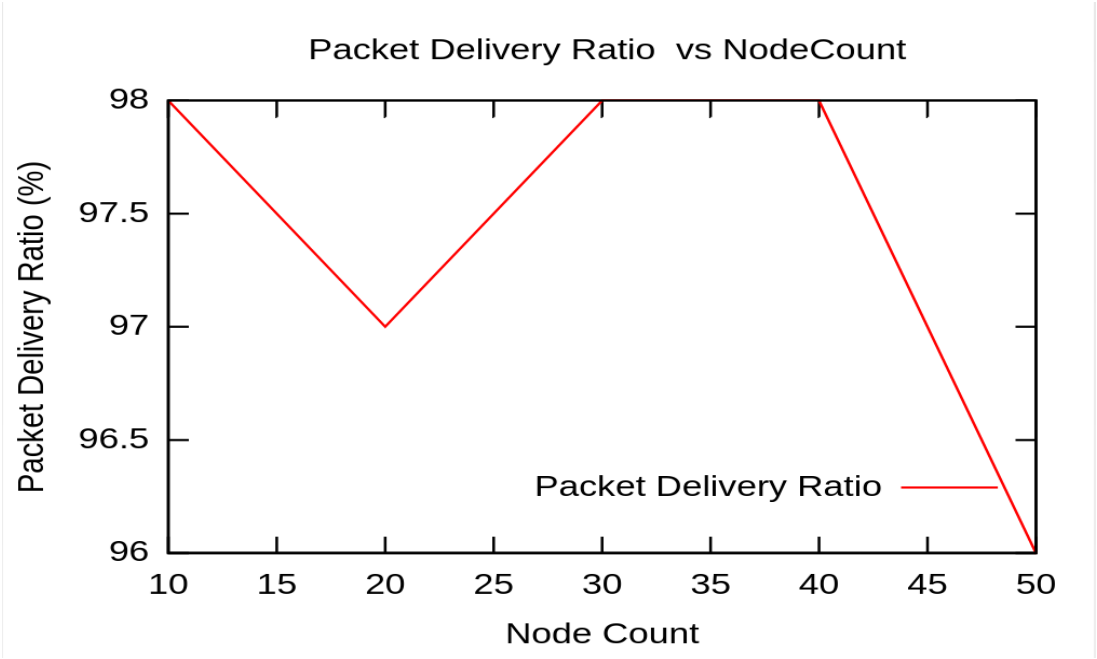
1.Throughput vs Node count



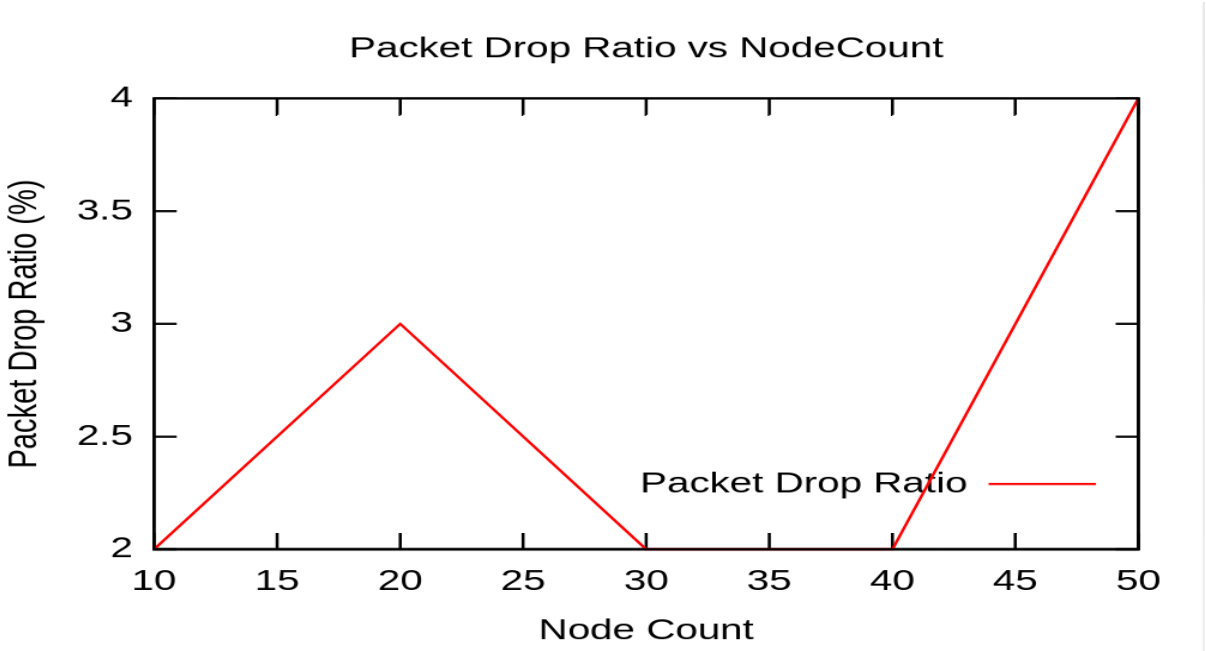
2.End-to-End delay vs Node count



3.Packet Delivery Ratio vs Node count



4.Packet Drop Ratio vs Node count



In all the cases number of packets were approximately 2000 , packet rate was 80 packets/sec and flow count was same as node count .

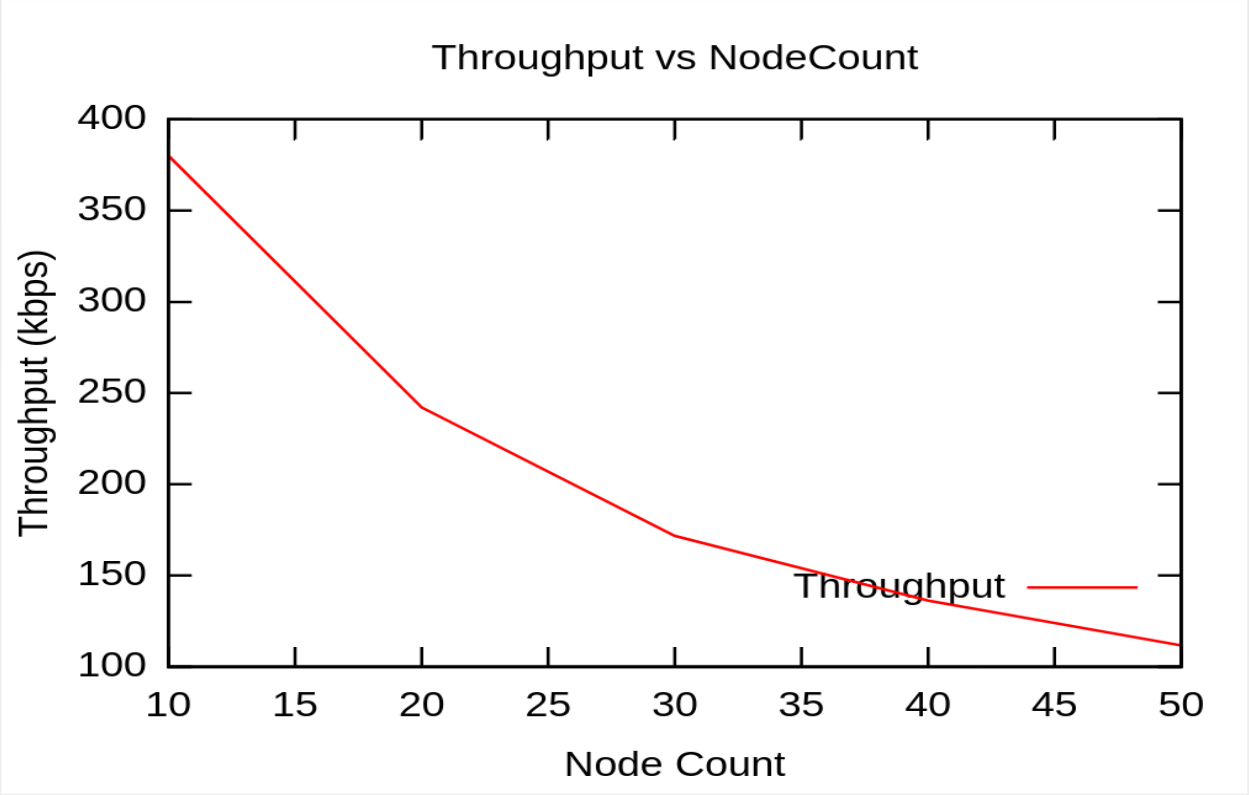
(LRWPAN, before modification)

In terms of lrwpan network a peculiar behavior was noticed and that is even if the node count, packets per second, coverage area ,mobility and other parameters were changed the network throughput always remained same and packet delivery ratio was 100% and packet drop ratio was 0%.

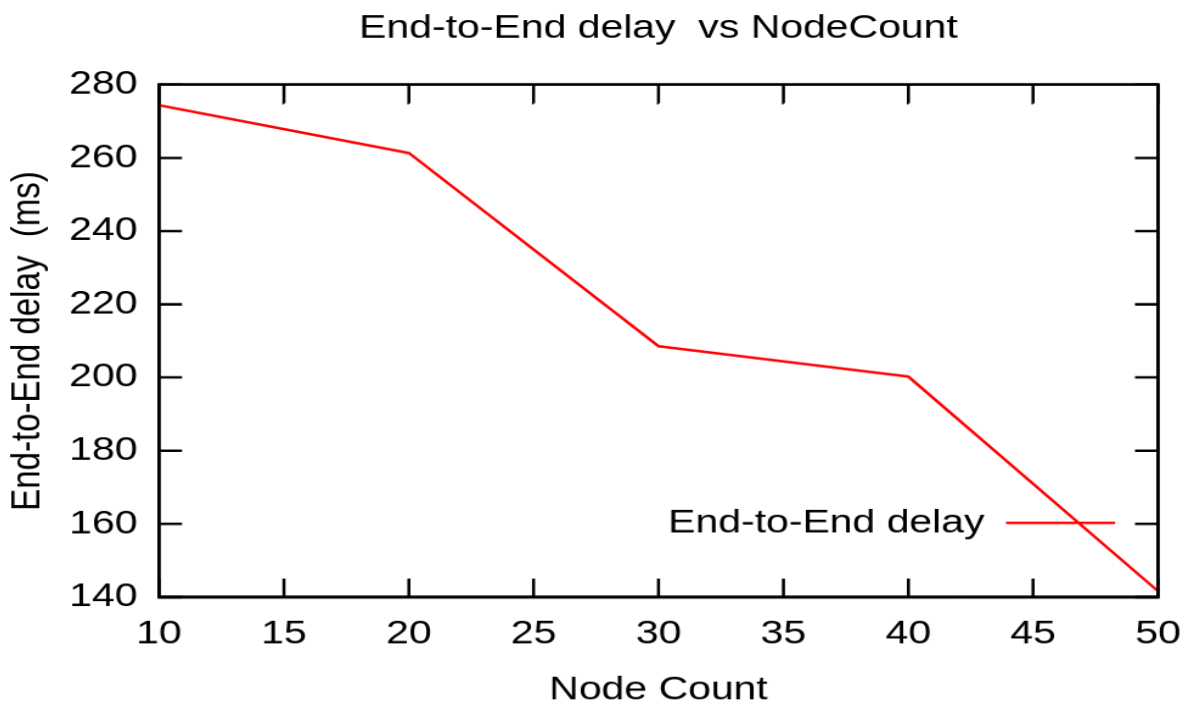
Graphs for Task B

(Wired, after modification)

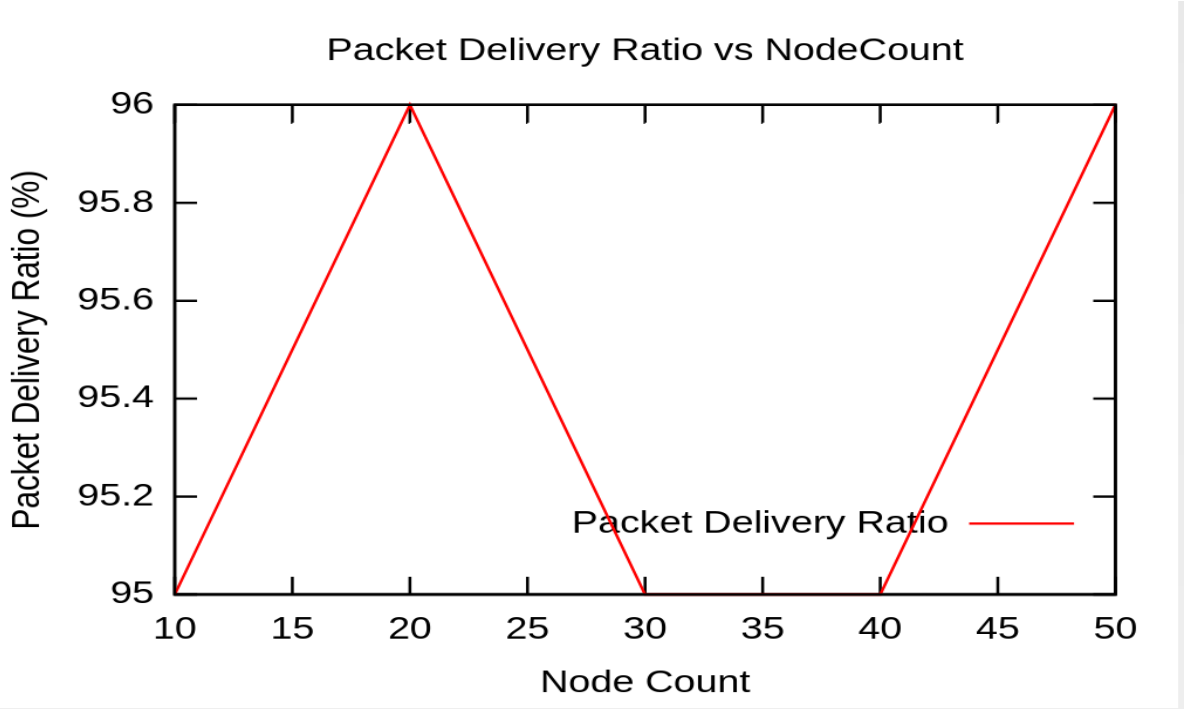
1.Throughput vs Node count



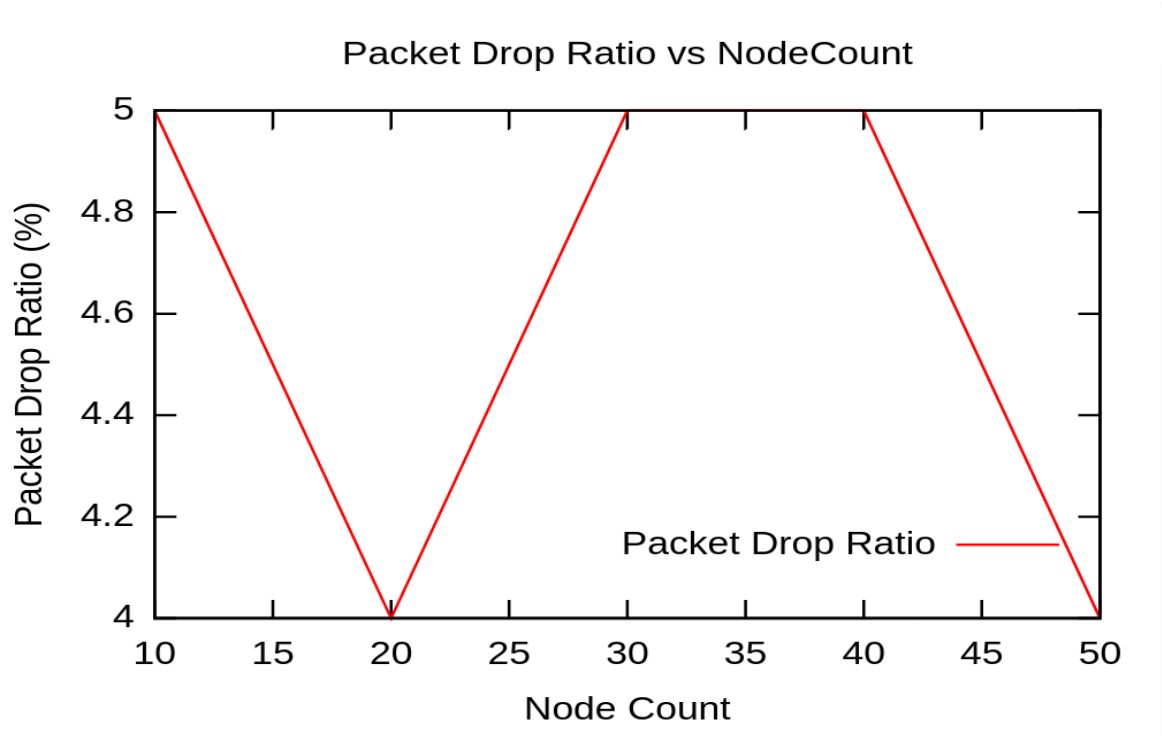
2.End-to-End delay vs Node count



3.Packet Delivery Ratio vs Node count



4.Packet Drop Ratio vs Node count



In all the cases number of packets were approximately 2000 , packet rate was 80 packets/sec and flow count was same as node count .

(LRWPAN, after modification)

In terms of lrwpan network a peculiar behavior was noticed and that is even if the node count, packets per second, coverage area, mobility and other parameters were changed the network throughput always remained same and packet delivery ratio was 100% and packet drop ratio was 0%.

Summary Findings

A significant outcome derived due to the modification of red queue sdisc in ns3. The proposed algorithm was designed to act as a shield to Ldos attacks. The outcomes show a positive tend toward our intention. After modification we got slight but negligible changes of the performance matrices. They are listed below.

1.Due to potential Ldos packet dropping Network Throughput was slightly decreased.

2.Due to extra checking overall End-to-End delay was slightly increased.

3.For similar reason as 1 Packet Delivery Ratio was merely decreased.

4.For similar reason as 1 Packet drop ratio was increased by a low margin.

So, the above outcomes basically proves that the algorithm was implemented successfully as the authors of the algorithm suggest that due to algorithm applied Network performance metrics nearly remains the same.