**Project Report**

**Network disruption analysis using network simulator NS2**

**ABSTRACT**

In a large computer network failure of one or several bilateral connections is a common occurrence. This nature of failure can either be related to the software or hardware glitches. This failure can lead to disruption in the flow of data within the network.

When the computer network detects any kind of failure in any of the nodes or connections in between, it tries to find an alternate path for the flow of data inside the network. There are several routing algorithms that can be implemented in the network to find optimal path or routes for data flow. Most of the routing algorithms treat the computer network structure as a graph. One of those algorithm is **Distance Vector Routing** algorithm. We applied this algorithm in our project to test its performance in an event of connection failure.

**NS2** or **Network Simulator 2** is a simulator built by DARPA under the VINT project.

We used this simulator to simulate our computer network built up of virtual nodes and connections. Also, we used multiple TCP and UDP sources along with several different congestion control algorithms like Reno, New Reno and Vegas to send data to their respective destination or sink.

Finally, we disabled different connection between nodes one at a time to simulate connection failure in a real computer network.

This project seeks to determine the performance of the traditional Distance vector routing algorithm applied inside a certain computer network topology when a connection failure event happens.

It is hoped that this project will help us understand how computer networks and routing algorithm behave in an event of connection failure and thus enhance our knowledge regarding network design.

**INTRODUCTION**

Computer networks are built up of several nodes and connections between them. These nodes can be a personal computer, router, switch etc. Real life computer networks like internet are vast and are built up of billions of computers.

A computer network is always vulnerable to a connection failure. In some cases these failures could be due to a malfunction node which leads to failure of all connections to and from it, while in some cases these connection failure occurs due to some software or hardware glitch in the connection equipment itself.

In an event of connection failure the routing algorithm governing the data flow inside the computer network try to pass the faulty connection or node and find an alternative path for data flow. This behavior leads to normal operations in a computer network even if parts of the network cannot be used for networking. Different routing algorithms use different approaches to find an alternate data flow path. Better algorithms lead to better performance in an event of some connection failure. Also, the performance of a network depends on its design as disruption of certain nodes or connections can lead to massive congestion inside the network eventually leading to packet loss and loss of performance.

**DESCRIPTION**

We will be monitoring the performance of a specific network topology. The network is built up of several nodes and connections. We disable connections one at a time and then observe the throughput performance of the network. We are using Distance Vector routing algorithm also known as Bellman-Ford algorithm for routing inside the network.

Adistance-vector routing (DVR) protocol requires that a router inform its neighbors of topology changes periodically. Each router maintains a Distance Vector table containing the distance between itself and all possible destination nodes. Distances, based on a chosen metric, are computed using information from the neighbors’ distance vectors.

Distance vector routing algorithm:-

1. A router transmits its distance vector to each of its neighbors in a routing packet.
2. Each router receives and saves the most recently received distance vector from each of its neighbors.
3. A router recalculates its distance vector when:
   * It receives a distance vector from a neighbor containing different information than before.
   * It discovers that a link to a neighbor has gone down.

When a node x receives new DV estimate from any neighbor v, it saves v’s distance vector and it updates its own DV using B-F equation:

dx(y) = min{ c(x,v) + dv(y) , dx(y)} for each node y in N

where ,

dx(y) = Estimate of least cost from x to y

c(x,v) = Node x knows cost to each neighbor v

dx = [dx(y): y ∈ N ] = Node x maintains distance vector

NS2 is used to simulate all the network components and algorithms. The simulator is capable of simulating various transport layer protocols like TCP and UDP as well as application layer protocols like FTP and CBR. The simulator also supports congestion algorithms like Reno, New Reno and Vegas. Routing algorithms are also implemented in the simulator. The simulator also implements queuing algorithms like SQF.

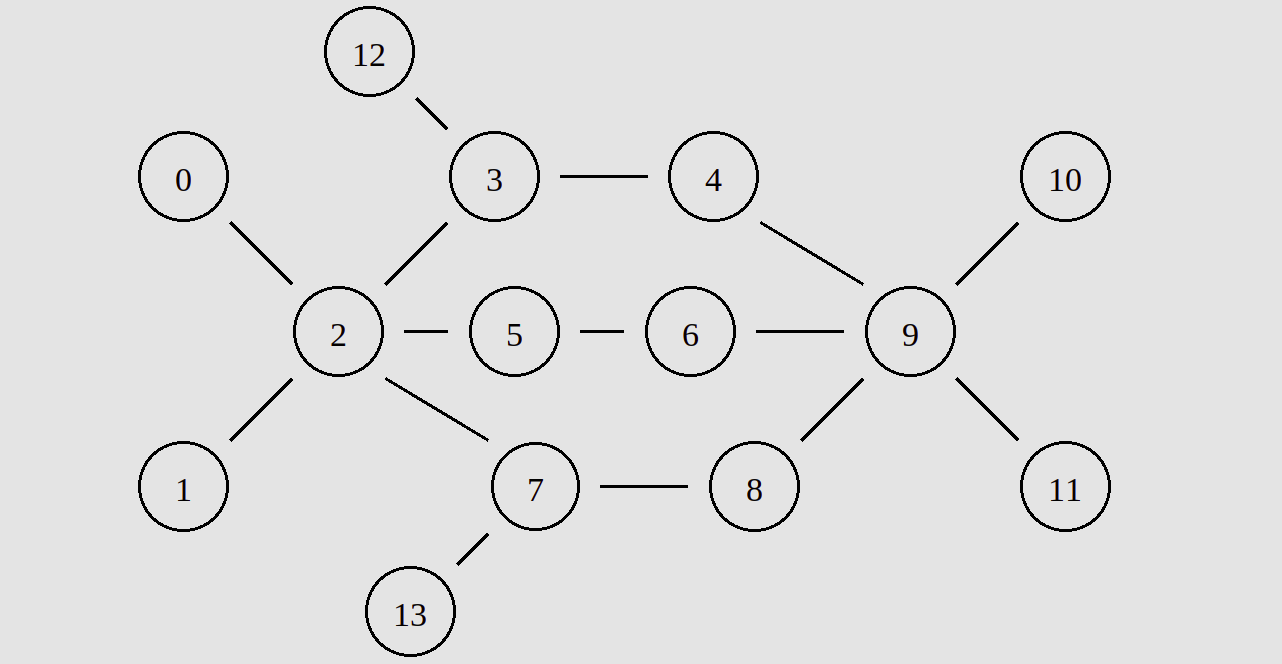
The various TCP source nodes apply various congestion control algorithms like Reno, New Reno, and Vegas etc.

We are using several different congestion control algorithms to observe

NS2 uses a module named NAM (Network Animator) to animate the entire simulated network. It represents every packet as a small solid square. Packets from different sources can be colored differently. NAM also animates dropped packets as falling packets.

**IMPLEMENTATION**

Below is the topology of the network we implemented



There are total 14 nodes in the topology out of which 4 are source nodes and 2 are destination nodes.

Node 0 UDP source

Node 13 UDP source

Node 1 TCP source

Node 12 TCP source

Node 10 UDP Sink (Loss Monitor)

Node 11 TCP Sink

The bandwidth and propagation delay of each connection in between these nodes is given below:-

Node 0 and 2 -> 4Mb 10ms

Node 1 and 2 -> 4Mb 10ms

Node 2 and 3 -> 3Mb 10ms

Node 3 and 4 -> 3Mb 10ms

Node 2 and 5 -> 2Mb 5ms

Node 5 and 6 -> 2Mb 5ms

Node 2 and 7 -> 4Mb 10ms

Node 7 and 8 -> 4Mb 10ms

Node 4 and 9 -> 4Mb 10ms

Node 6 and 9 -> 4Mb 10ms

Node 8 and 9 -> 4Mb 10ms

Node 9 and 10 -> 4Mb 10ms

Node 9 and 11 -> 4Mb 10ms

Node 3 and 12 -> 4Mb 5ms

Node 13 and 7 -> 4Mb 5ms

Since the distance between Node 2 and Node 5 is minimum thus DV algorithm chooses path passing through node 2 - node 5. Despite the existence of node 2 - node 3 and node 2 - node 7 i.e. 2 paths of higher bandwidth. Since path with lower bandwidth is chosen we observe packet drops at node 3.

Network disruption analysis:

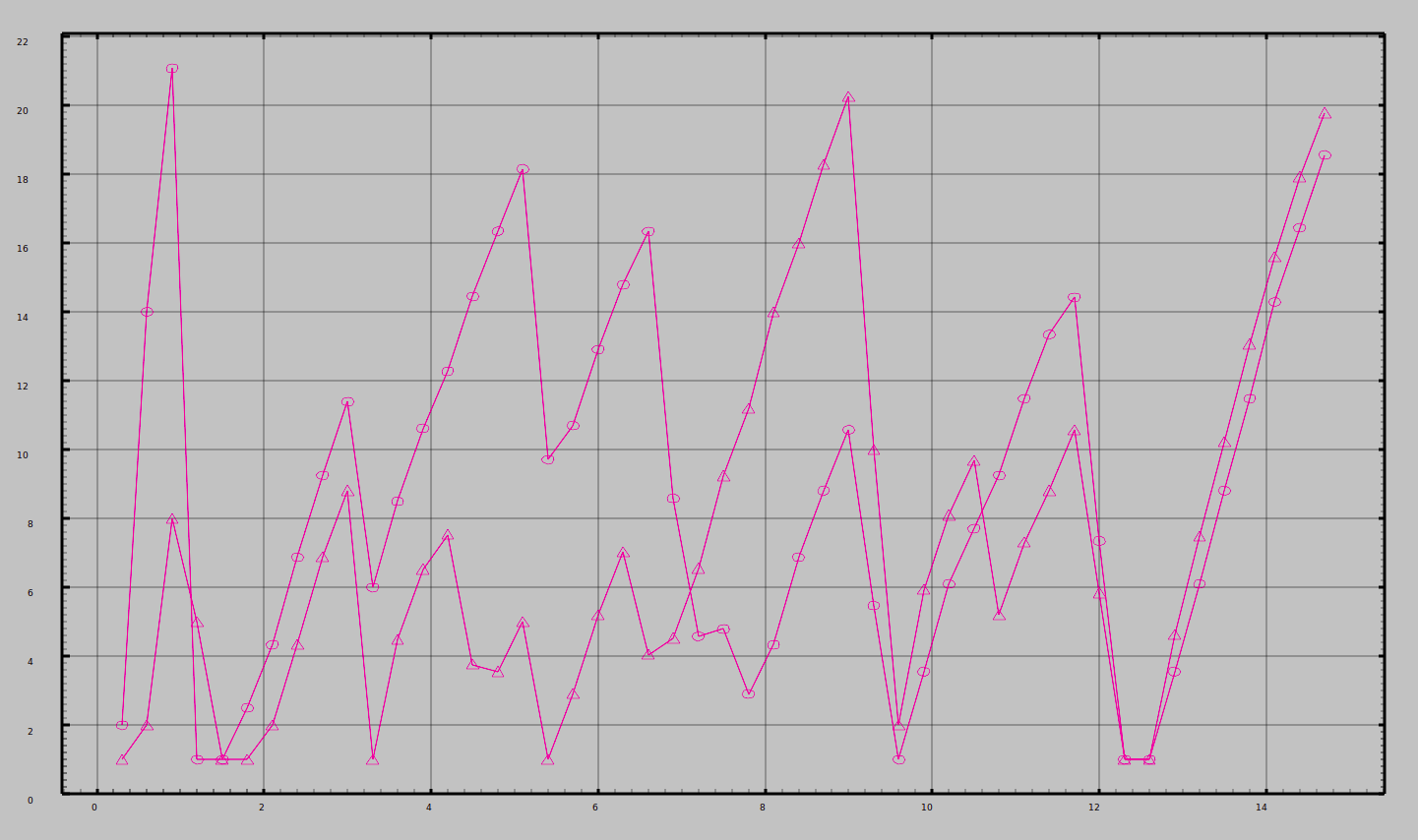
We implemented two nested for loops and try to disrupt link between every two nodes. If two nodes are not adjacent we catch the exception and pass the loop to the next node.

Link is disrupted and restored at an interval of **0.5 sec**. And we analyze the congestion window for TCP nodes and bandwidth received by UDP sink at an interval of **0.7 sec.** That is we analyze at mean of disrupted or restored state.

Difference between New Reno, Reno, Vegas:

We implemented a pair of above mentioned algorithms on node 1 under same topology circumstances, and following graphs were observed:

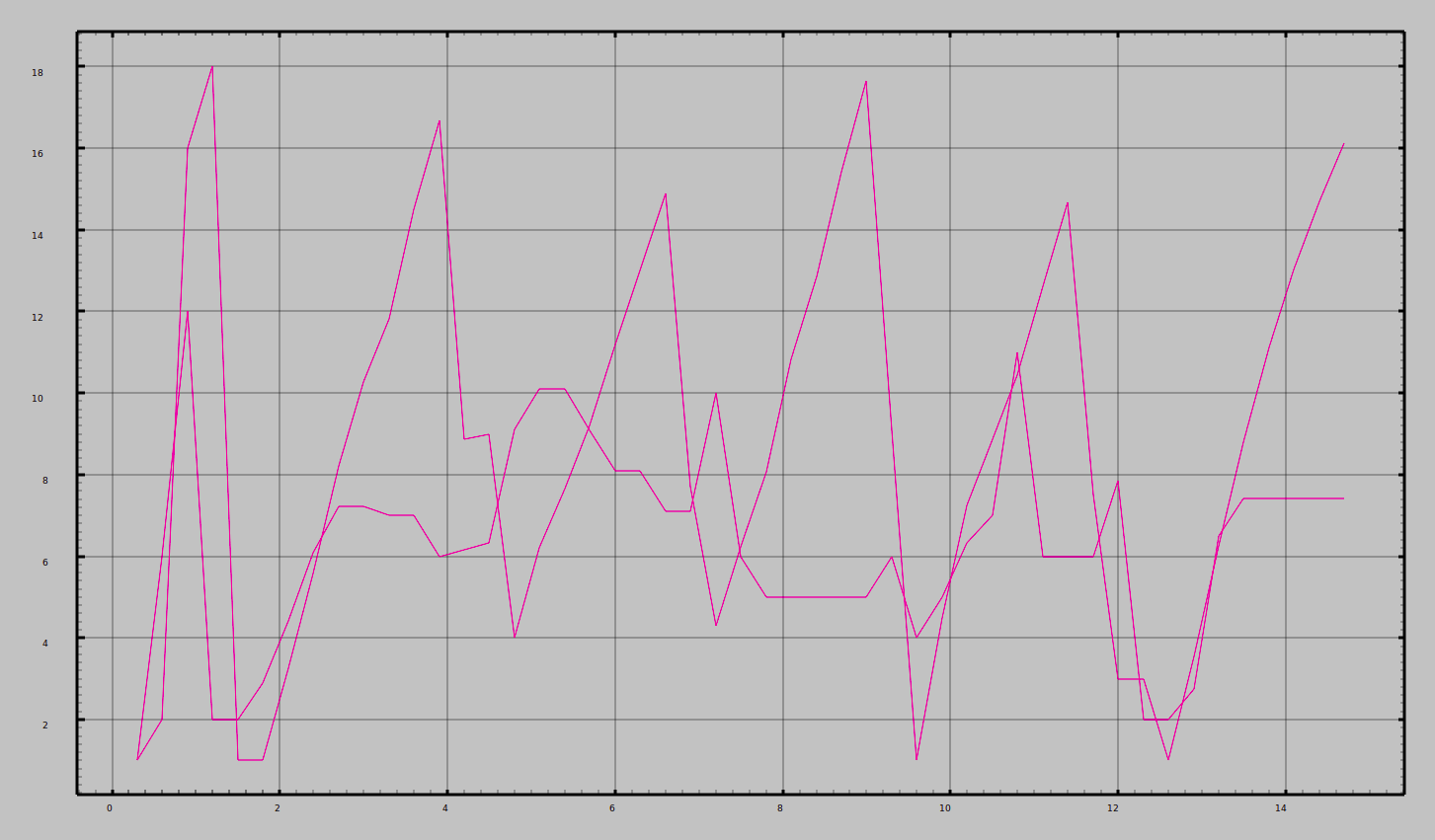
**Reno vs New Reno:**



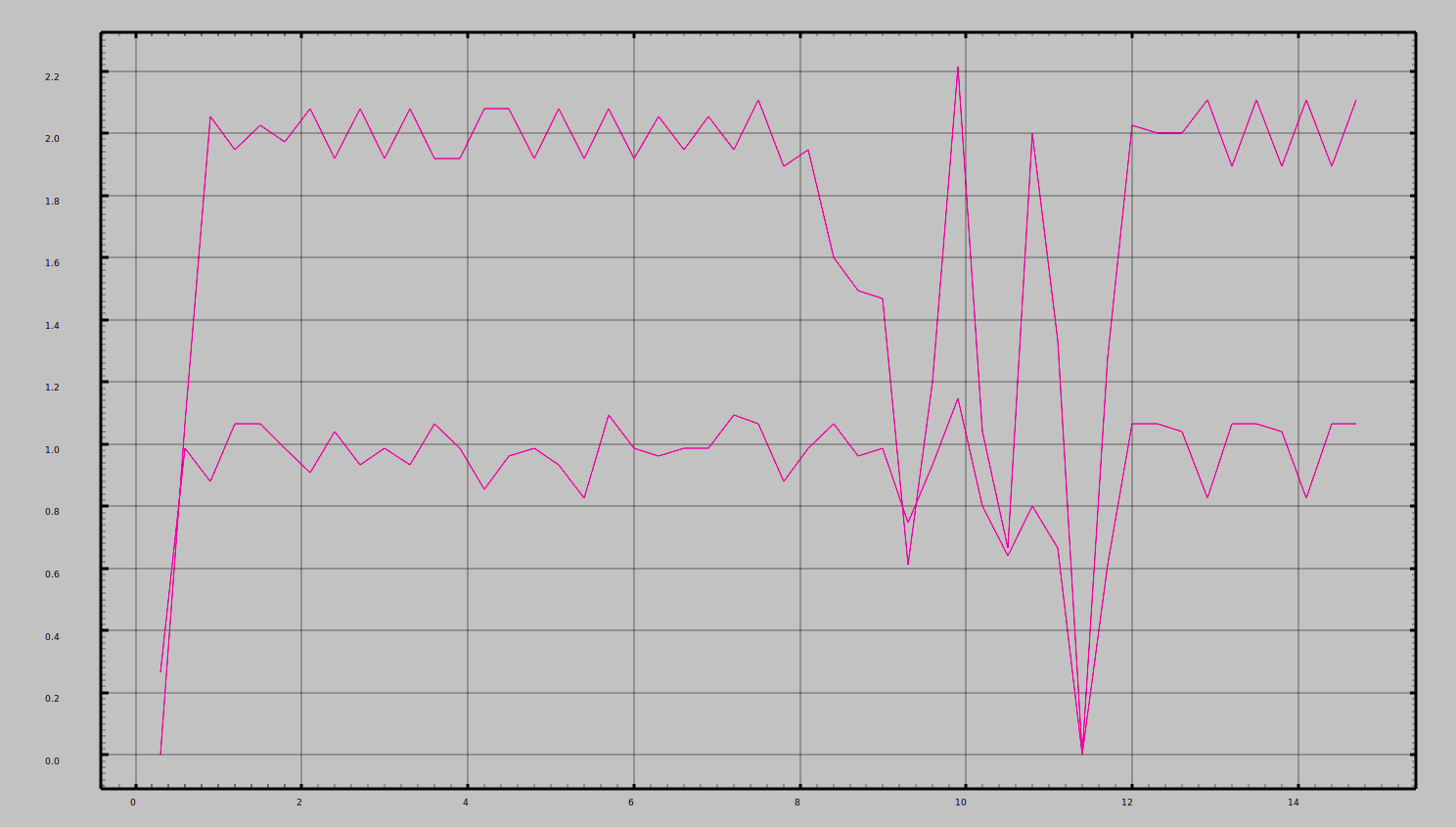
**Reno vs Vegas:**



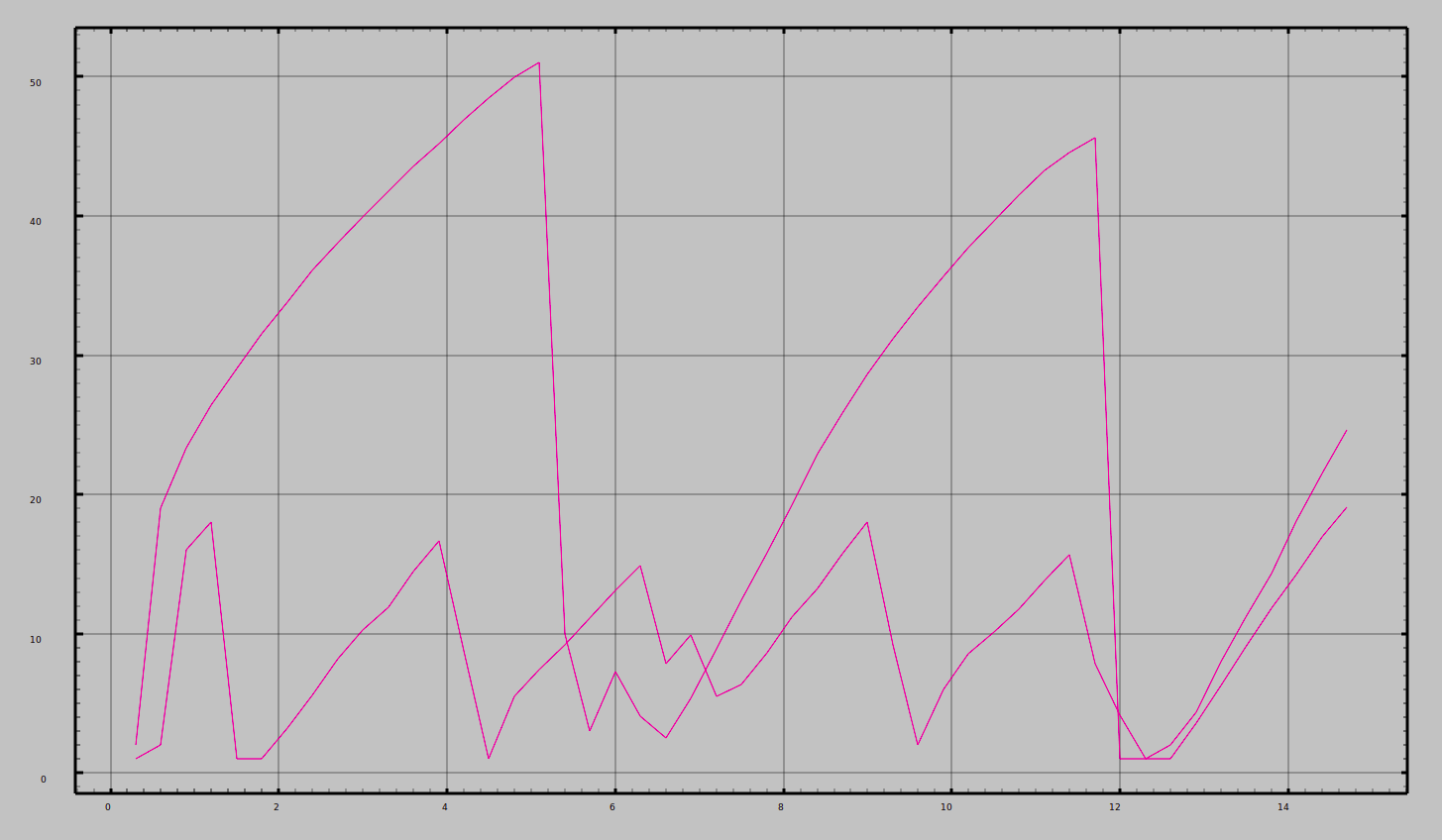
**New Reno vs Vegas:**



**UDP Bandwidth analysis:**



Further we analyze two TCP nodes set at alternate position to observe the working of Distance Vector algorithm and plot the following graph:



**RESULTS and CONCLUSION:**

1. We observed that TCP stops sending packets during congestion while UDP mindlessly keeps sending packets into the network. This proves TCP implements congestion control.
2. We found that **New Reno performs best** for our topology out of other congestion control algorithms. As observed from the graph.
3. Same TCP congestion control algorithm when implemented by different TCP nodes perform very differently as distance vector algorithm chooses different paths for different nodes.
4. Weakest link in our topology was link from node 9 - node 10 and node 9 - node 11. As observed from the graph. The minimum throughput is observed at **11.8 sec.**
5. Maximum throughput of the network is 2.2 Mbps achieved by CBR packets operating on UDP protocol.