CONGESTION CONTROL SIMULATOR

Pankaj Gupta -061319

May-2010

Submitted in partial fulfillment of the degree of

Bachelor of Technology

In

Computer Science & Engineering

DEPARTMENT OF COMPUTER SCIENCE AND

ENGINEERING AND INFORMATION TECHNOLOGY

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,

WAKNAGHAT, SOLAN-173215, INDIA

CERTIFICATE

This is to certify that the work entitled, "Congestion Control Simulator" submitted by **Pankaj Gupta** in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science and Engineering of Jaypee University of Information Technology has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

PROJECT SUPERVISOR

Mr. Sanjeev Patel

Lecturer,

Department of CSE & IT, JUIT

Brig (Retd.) S.P. Ghrera

Head of the Department

Department of CSE & IT, JUIT

ACKNOLEDGEMENT

Computer Science is fast developing subject & new dimensions are being added from time to time. During the last decade many new dimensions have been introduced and many old ones have been redefined. In light of new development and recent findings, I devote the task that was asked from me during my 4th year at Jaypee University of Information and Technology to Congestion Control Simulator and to the needs of computer science.

I express my sincere gratitude and thanks to all those who have helped me in the progress of the project. Of all the persons who have helped me, I would first of all like to thank **Mr. Sanjeev Patel,** under whose able guidance we have worked on our project.

I would also like to extend my gratitude to **Brig. S.P. Ghrera**, Head of Department, Computer Science, JUIT.

Pankaj Gupta Roll no. 061319

Pankaj

B.Tech (CSE)

Jaypee University of Information and Technology

Abstract

Network technologies have been revolutionized in the last decade. The speed and capacity of various components in a networking system, such as transmission media, switches, and routers have been drastically increased. Millions of users and billions of traffic have introduced into networks. So the congestion is also exponentially increasing.

To control that congestion, one of the major components in a QoS-enabled network is active queue management (AQM). Over the last decade numerous AQM schemes have been proposed in the literature. However, much recent work has focused on improving AQM performance through alternate approaches. This study focuses on an unbiased comparative evaluation of the various proposals. The evaluation methodology uses the queue, delay, and loss characteristics – a subset of network characteristics that can be used to describe the behavior of network entities, and give their mathematical description. Since each of the AQM has distinct features so to test them in different scenario is became necessary so that they can be deployed in real scenario. To test them in different environment in NS2 is very tedious task. So the tool will reduce the complexity of computing the performance in different scenario.

Table of Contents

List of Figures

Definition and Acronyms

TCP/IP:- stands for "Transmission Control Protocol". It provides a reliable, ordered delivery of a stream of bytes from a program on one computer to another program on another computer.

UDP: - stands for "User Datagram Protocol". It provides an unreliable service and datagrams may arrive out of order, appear duplicated, or go missing without notice.

FTP: - stands for "File Transfer protocol". It is a standard network protocol used to exchange and manipulate files over an Internet Protocol computer network, such as the Internet.

CBR :- Constant Bit Rate

RED :- Random Early Detection

SRED:- Stabilized Early Detection

FRED:- Flow Random Early Detection

SFQ:- Stochastic Fair Queuing

Chapter 1 INTRODUCTION

1.1 What is Congestion?

The robustness of today's Internet depends heavily on the TCP congestion control mechanism. However, as more and more UDP applications (e.g. packet audio/video applications) are deployed on the Internet, people cannot rely on end users to incorporate proper congestion control. Router mechanisms must be provided to protect responsive flows from non-responsive ones, and prevent "Internet meltdown". Several methods have been proposed for the management of shared resources on the Internet, active queue management is one of the major approaches.

As Internet can be considered as a ***Queue of packets,*** which transmitting nodes constantly add packets into the queue, receiving nodes constantly remove packets from the queue. So, in a situation where too many packets are present in this queue, such that rate of sending the packets exceeds the receiving rate, such a situation is termed as ***Congestion***.

Main reason of network congestion: more number of packets into the network than it can handle

Objective of congestion control: maintain the number of packets in the network below the level at which performance falls off dramatically.

The nature of a Packet switching network can be summarized in following points:

- A network of queues
- At each node, there is a queue of packets for each outgoing channel
- If packet arrival rate exceeds the packet transmission rate, the queue size grows without bound
- When the line for which packets are queuing becomes more than 80% utilized, the queue length grows alarmingly

1.2 Causes of Congestion

Insufficient memory to hold these packets at the router.

Slow processors lead to build up of queue even if there is excess of line capacity.

Low-Bandwidth lines can also cause congestion.

Congestion tends to grow itself due to retransmission

bursty **nature of traffic**

1.3 Effects of Congestion

Throughput and Delay are two vital parameters for network performance and is most affected by the congestion. Throughput can be defined as the percentage utilization of the network capacity. Delay can be time as time reach at receiver end .

Fig. 1 shows how throughput is affected as offered load increases.

The delay also increases with offered load, as shown in Fig. 2.

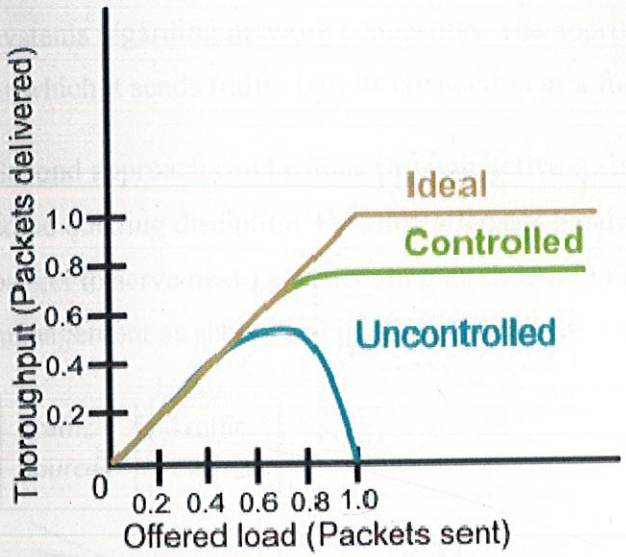


Fig. 1 Effect of congestion on throughput

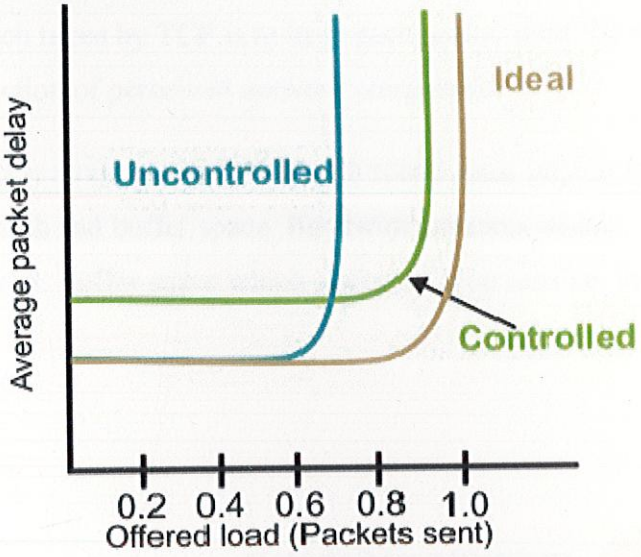


Fig. 2 Effect of congestion on delay

1.4 Flow Control versus Congestion control

Flow control regulates the transmission of data between devices, but it considers what is going on within each of the devices on the connection, and not what is happening in devices between them. It relates to the point-point traffic between a given sender and a receiver. Flow control always involves some kind of feedback from receiver to sender to tell sender how things are at other end of the network.

In practice, what is going on at layer three can be quite important. Considered from an abstract point of view, our server and client may be connected "directly" using TCP, but all the packets we transmit are carried across an internet and routers between different networks. These networks and routers are also carrying data from many other connections and higher-layer protocols. If the internet becomes very busy, the speed at which segments are carried between the endpoints of our connection will be reduced,

and they could even be dropped. This is called *congestion control*. Congestion control has to do with making sure that subnet carry the offered traffic. It is the global issue, involving the behavior of all the hosts, router, link, store and forward mechanism between them in the entire subnet or internet.

1.5 Mechanism for controlling the congestion

The two basic mechanisms of congestion control are:

- One is recovery from congestion, when congestion has already taken place
- Another is preventive, where precautions are taken so that congestion cannot occur.

First approach can said as **TCP Congestion Control.** TCP must use end-to-end congestion control rather than network-assisted congestion control, since the IP layer provides no feedback to the end systems regarding network congestion. The approach taken by TCP is to have each sender limit the rate at which it sends traffic into its connection as a function of perceived network congestion.

Second approach can be done through **active queue management** .In this each router must implement some queuing discipline. Queuing allocates bandwidth and buffer space. Bandwidth ensures which packet to serve next i.e. scheduling as shown in Fig. 3. Buffer space which packet to drop next i.e. buffer management as shown in Fig. 4.

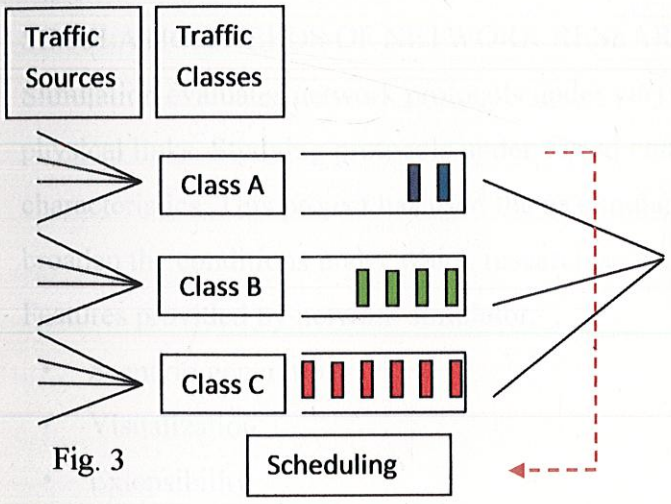


Fig. 3

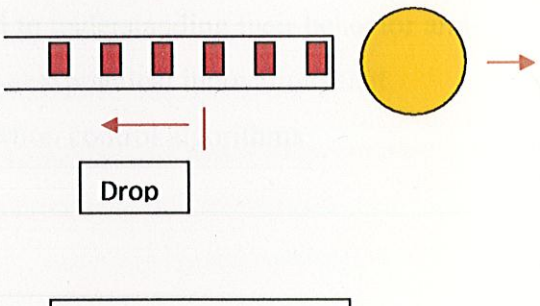


Fig. 4

Chapter 2 Project Objective and Approach

2.1 PROBLEM STATEMENT

To develop a tool for determining the performance of various congestion control algorithms for queue management at the router in a network. The ns network simulator is a common tool frequently used, however writing long ns scripts is not feasible and comparing their result in graph is very tedious work. We used a custom-made visualization tool where topologies are created using a mouse and parameter is directly filled using interface forms. The Network Simulator 2 (NS-2) is a popular and powerful simulation environment, and the number of NS-2 users has increased greatly in recent years.

2.2 OBJECTIVE

SIMULATION NEEDS OF NETWORK RESEARCHERS

Simulation evaluates network protocols under varying network conditions virtually without setting physical links. Studying protocols under varied conditions is critical to understanding their behavior and characteristics. This project has used the ns simulator to provide several practical innovations that broaden the conditions under which researchers can evaluate congestion control algorithms.

Features provided by network simulator:

- Scenario generation
- Visualization
- Extensibility
- Analyzing

2.3 SCOPE OF THE PROJECT

Traffic on the Internet tends to fluctuate and to be greedy. Ideally, a router queue management algorithm should allow temporary bursty traffic, and penalize flows that persistently overuse bandwidth. Also, the algorithm should prevent high delay by restricting the queue length, avoid underutilization by allowing

temporary queuing, and allocate resource fairly among different types of traffic. In practice, most of the routers being deployed use simplistic Drop Tail algorithm, which is simple to implement with minimal computation overhead, but provides unsatisfactory performance.

To attack this problem, many queue management algorithms are proposed, such as RED, FRED, BLUE, REM, SRED and SFQ from which in the real scenario RED is implemented, rest are being under research.

Our project will develop a tool that can be used to compare these various congestion control algorithms in any number of bottleneck links. They can compare their performance on the basis of the performance matrices. We can evaluate several queue management algorithms with respect to their abilities of maintaining high resource utilization, identifying and restricting disproportionate bandwidth usage, and their complexity. And also new developed algorithm can also append in ns2 and can be compared. Now we only compare the performance of FRED, BLUE, SFS, REM, SRED, RED and Drop Tail as the evaluation baseline. The driving motivation for the development of the tool is to make these functions user friendly through graphical interfacing. It will generate graph on the basis of the network created using the interface.

2.4 Advantages

We used a custom-made visualization tool i.e. Network Simulation by using mouse where topologies are created using a mouse and then the TCL script is generated automatically. The tool is interactive and has an easy interface to create topologies consists very large network nodes and define agents and protocols that govern the communication setup. The graph is generated automatically, the complexity also do not exceed O(n), reads the output transcript only one times for it. And the simulation uses ns2 and nam for simulation and visualization.

2.5 Limitations

The analysis results in the project are completely based on the virtual simulation data. The exact performance measures in the real life scenarios where not measured. The parameter settings are limited to node, agent and applications. Others protocols need to be sets in TCL script manually only thus total free from TCL coding is not achieved.

Chapter 3 Background Study

3.1 Congestion Control

The objective of congestion Control is to maintain the number of packets within the network below the level at which performance falls off dramatically. At the broadest level, we can distinguish among congestion control approaches based on the whether or not the network layer provides any explicit assistance to the transport layer for congestion control purposes:

- **End-end congestion control.** In this approach, the network layer provides *no explicit support* to the transport layer for congestion control purposes. If there is congestion in the network then the information to sender will be indicated either by the packet loss or delay.TCP takes this approach to control the network congestion, since IP protocol fails to provide information to the end systems. The loss of TCP packet (as indicated by a timeout or a triple duplicate acknowledgement) is taken as an indication of network congestion and TCP decreases its window size accordingly. TCP also uses increasing round-trip delay values as indicators of increased network congestion. But if there is congestion in the network then at router level the TCP protocols continue to send the packets unless and until informed the other end or loss of packets. Although it's an effective way but congestion control can't really totally on it.

- **Network-assisted congestion control.** With network-assisted congestion control, network-layer components (i.e., routers) provide explicit feedback to the sender regarding the congestion state in the network. This feedback may be as simple as a single bit indicating congestion at a link. So the flooding and dropping at the router level can be decreased. It's an preventive approaches to avoid the congestion. At router there is some regulator which continues to record the flow of packet if it sees the misuse or growing of traffic it then indicate the sender or receiver end to stop or slow the transmission of the packet.

3.1.1 TCP Congestion Control

- TCP must use end-to-end congestion control rather than network-assisted congestion control, since the IP layer provides no feedback to the end systems regarding network congestion.
- The approach taken by TCP is to have each sender limit the rate at which it sends traffic into its connection as a function of perceived network congestion.
- If a TCP sender perceives that there is a little congestion on the path b/w itself & the destination then the TCP sender increases the sending rate. And if the sender perceives that there is congestion along the network path, hence the sender reduces its send rate.
- The variation in the send rate is achieved by adjusting the value of congestion window(CongWin).

LastByteSent – LastByteAcked <= min{CongWin, RcvWin}

TCP Congestion Control

- Three major components:
  - Additive-increase, multiplicative-decrease
  - Slow start
  - Reaction to timeout events

In summary:

- When the congestion window is below the threshold, the congestion window grows exponentially.
- When the congestion window is above the threshold, the congestion window grows linearly.
- Whenever there is a timeout, the threshold is set to one half of the current congestion window and the congestion window is then set to one.

3.1.2 Network Congestion Control

In this type **Active Queue Management (AQM)** is Explicit Feedback that is used to control or manages the congestion in networks. AQM employs a single Explicit Congestion Notification (ECN) bit in a packet header to feed back the congestion that occur in intermediate devices such as routers and gateways to the end users or end nodes. These intermediate devices or gateways transfer all type of data from one host computer to another. The gateway will mark packets if end host computers support ECN; otherwise it will drop the packets during congestion. There are two big advantages of using explicit feedback the first being that it is a fast method of notifying sources in order to change the state of congestion as soon as possible and the second is that congestion normally occurs in the bottleneck router so to have a congestion feedback from routers means that the congestion information is precise. Many AQM systems have been introduced recently in the literature that aims to improve the QoS that the internet users demand. One well known AQM algorithm is random early detections (RED) which relies on queue thresholds on its operation. There are many developments of RED such as REM, SRED and FRED among others. These however rely on static thresholds which can be restrictive when parameters such as the arrival rate changes. There are hundreds of algorithms that are developed for congestion control at the router level , each controlling the certain aspect of the congestion. These can classified on basis of various parameters as shown in Fig. 5.

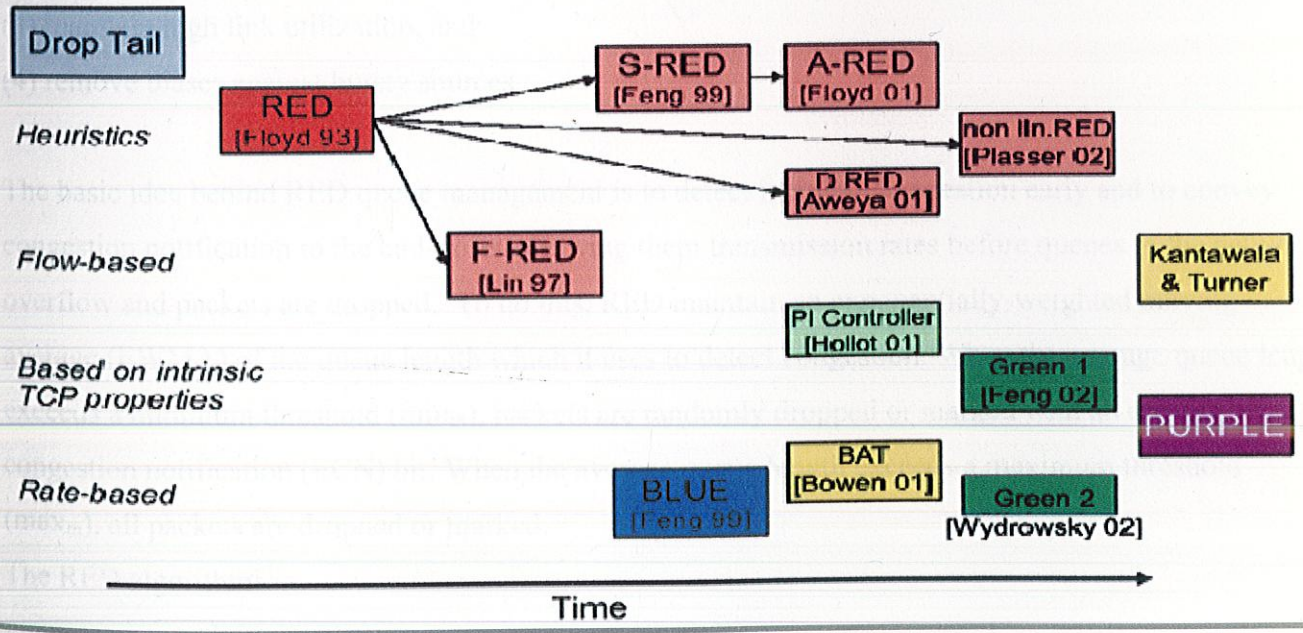


Fig. 5 Classification of AQM based Algorithm

The various algorithms studied during the project are given below:

3.1.2.1 Drop Tail

Drop Tail is a simple queue management algorithm used by Internet routers t decide when to drop the packets. Once a queue has been filled, the router begins discarding all additional datagrams and dropping the tail of the sequence of it.

Drop-tail issues: Routers are forced to have large queues to maintain high utilizations. Larger buffers will lead to larger steady state queues and delays.

Synchronization: end hosts react to same events because packets tend to be lost in bursts.

Lock-out: a side effect of burst times and synchronization is that a few flows can monopolize queue space.

3.1.2.2 RED (Random Early Detection)

RED was designed with the objectives to

(1) minimize packet loss and queuing delay,

(2) avoid global synchronization of sources,

(3) maintain high link utilization, and

(4) remove biases against bursty sources.

The basic idea behind RED queue management is to detect incipient congestion early and to convey congestion notification to the end-hosts, allowing them transmission rates before queues in the network overflow and packets are dropped. To do this, RED maintain an exponentially-weighted moving average (EWMA) of the queue length which it uses to detect congestion. When the average queue length exceeds a minimum threshold (min_{th}), packets are randomly dropped or marked with an explicit congestion notification (ECN) bit. When the average queue length exceeds a maximum threshold (max_{th}), all packets are dropped or marked.

The RED algorithm:

Initialization:

avg ← 0

count ← -1

for each packet arrival

calculate new avg. queue size avg:

if queue is nonempty

$$avg = (1 - w_q)avg + w_w q$$

else

$$m = f(time - q\_time)$$

$$avg = (1 - w_q)^m avg$$

if $min_{th} \leq avg \leq max_{th}$

increment count

calculate probability p_a:

$$p_b = max_p \frac{(avg - min_{th})}{(max_{th} - min_{th})}$$

$$p_a = \frac{p_b}{(1 - count.p_b)}$$

with probability p_a:

mark the arriving packet

count =0

elseif $max_{th} \leq avg$

mark the arriving packet

count =0

else

count = -1

when queue becomes empty

q_time = time

Saved Variables:

avg: average queue size

q_time: start of the queue idle time

count: packets since last marked pkt

Fixed parameters:

w_q: queue weight

: minimum threshold for queue

: maximum threshold for queue

: maximum value for

Other:

: current pkt-marking probability

q : current queue size

time: current time

f(t):a linear function of the time t

Now the Fig. 6 shows how the average queue length and its marking probability varies in between the minimum threshold and maximum threshold as the packet at the routers goes on increasing.

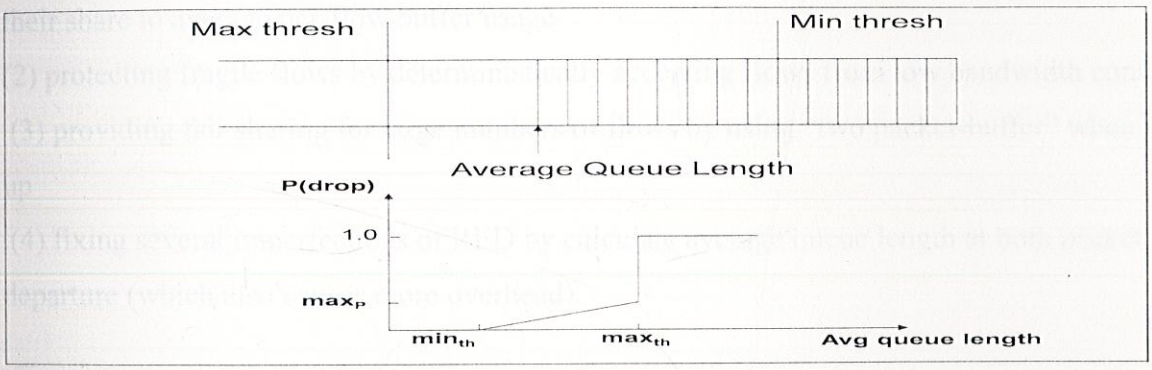


Fig. 6 RED policy

While RED is certainly an improvement over traditional droptail queues, it has several shortcomings. One of the fundamental problems with RED is that they rely on queue length as an estimator of congestion. While the presence of a persistent queue indicates congestion, its length gives very little information as to the severity of congestion. That is, the number of competing connections sharing the link. In a busy period, a single source transmitting at a rate greater than the bottleneck link capacity can cause a queue to build up just as easily as a large number of sources can. Since the RED algorithm relies on queue lengths, it has an inherent problem in determining the severity of congestion. As a result, RED requires a wide range of parameters to operate correctly under different congestion scenarios. While RED can achieve an ideal operating point, it can only do so when it has a sufficient amount of buffer space and is correctly parameterized.

3.1.2.3 FRED (Flow Random Early Drop)

Flow Random Early Drop (FRED) is a modified version of RED, which uses per-active-flow accounting to make different dropping decisions for connections with different bandwidth usages. FRED only keeps track of flows that have packets in the buffer, thus the cost of FRED is proportional to the buffer size and independent of the total flow numbers (including the short lived and idle flows). FRED can achieve the benefits of per-flow queuing and round-robin scheduling with substantially less complexity.

Some other interesting features of FRED include:
(1) penalizing non-adaptive flows by imposing a maximum number of buffered packets, and surpassing their share to average per-flow buffer usage
(2) protecting fragile flows by deterministically accepting flows from low bandwidth connections
(3) providing fair sharing for large numbers of flows by using "two packet-buffer" when buffer is used up
(4) fixing several imperfections of RED by calculate average queue length at both packet arrival and departure (which also causes more overhead).

Two parameters are introduced into FRED: *minq* and *maxq*, which are minimum and maximum numbers of packets that each flow is allow to buffer. In order to track the average peractive-flow buffer usage, FRED uses a global variable *avgcq* to estimate it. It maintains the number of active flows, and for each of them, FRED maintains a count of buffer packets, *qlen*, and a count of times when the flow is not responsive (*qlen* > *maxq*). FRED will penalize flows with high strike values. FRED processes arriving packets using the following algorithm(Fig. 7):

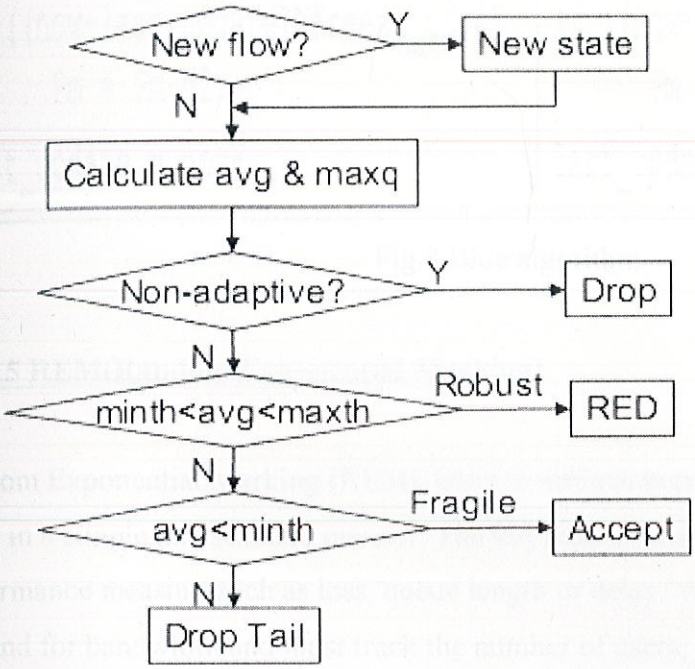


Fig. 7 FRED algorithm

3.1.2.4 BLUE

BLUE is an active queue management algorithm to manage congestion control by packet loss and link utilization history instead of queue occupancy. BLUE maintains a single probability, *Pm*, to mark (or drop) packets. If the queue is continually dropping packets due to buffer overflow, BLUE increases *Pm*, thus increasing the rate at which it sends back congestion notification or dropping packets. Conversely, if the queue becomes empty or if the link is idle, BLUE decreases its marking probability. This effectively allows BLUE to "learn" the correct rate it needs to send back congestion notification or dropping packets.

The typical parameters of BLUE are *d1*, *d2*, and *freeze_time*. *d1* determines the amount by which *Pm* is increased when the queue overflows, while *d2* determines the amount by which *Pm* is decreased when the link is idle. *freeze_time* is an important parameter that determines the minimum time interval between two successive updates of *Pm*. This allows the changes in the marking probability to take effect before the value is updated again. Based on those parameters. The basic blue algorithms can be summarized as following (Fig. 8):

Upon link idle event:	Upon packet loss event:

if ((now-last_update)>freeze_time)
Pm = Pm-d2;
Last_update = now;
	if ((now-last_updatte)>freeze_time)
	Pm = Pm+d1;
	last_update = now;

Fig.8 Blue algorithm

3.1.2.5 REM(Random Exponential Marking)

Random Exponential Marking (REM), aims to achieve both high utilization and negligible loss and delay in a simple and scalable manner. The key idea is to decouple congestion measure from performance measure such as loss, queue length or delay. While congestion measure indicates excess demand for bandwidth and must track the number of users, performance measure should be stabilized around their targets independently of the number of users. We explain the design rationale behind REM and present simulation results of its performance in wire line and wireless networks.

REM differs from RED only in the first two design questions: it uses a different definition of congestion measure and different marking probability function.

1. match rate clear buffer

It attempts to match user rates to network capacity while clearing buffers (or stabilize queues around a small target), regardless of the number of users.

2. sum prices

The end-to-end marking (or dropping) probability observed by a user depends in a simple and precise manner on the sum of link prices (congestion measures), summed over all the routers in the path of the user.

First, REM uses only local and aggregate information – in particular no per-flow information is needed – and works with any work conserving service discipline.

It updates its price independently of other queues or routers. Hence its complexity is independent of the number of users or the size of the network or its capacity.

Second, it is usually easier to sample queue length than rate in practice. When the target queue length b* is nonzero, we can bypass the measurement of rate mismatch $x_l(t) - c_l(t)$ n the price update .

Notice that $x_l(t) - c_l(t)$ is the rate at which the queue length grows when the buffer is nonempty. Hence we can approximate this term by the change in backlog, $b_l(t+1) - b_l(t)$. Then the update rule becomes:

$$pl(t+1) = \max(0,\ pl(t) + \gamma(\alpha l(ql(t) - qref) + xl(t) - cl(t)))$$

i.e., the price is updated based *only* on the current and previous queue lengths. ". It maintains "price" as a congestion measure to determine the marking probability, *pl*, based on rate mismatch (i.e., difference between input rate and link capacity) and queue mismatch (i.e., difference between queue length and target value q_{ref}. The cost is directly proportional to the queue occupancy.

Or it can simply be compared to the marking probability of the RED (Fig. 9):

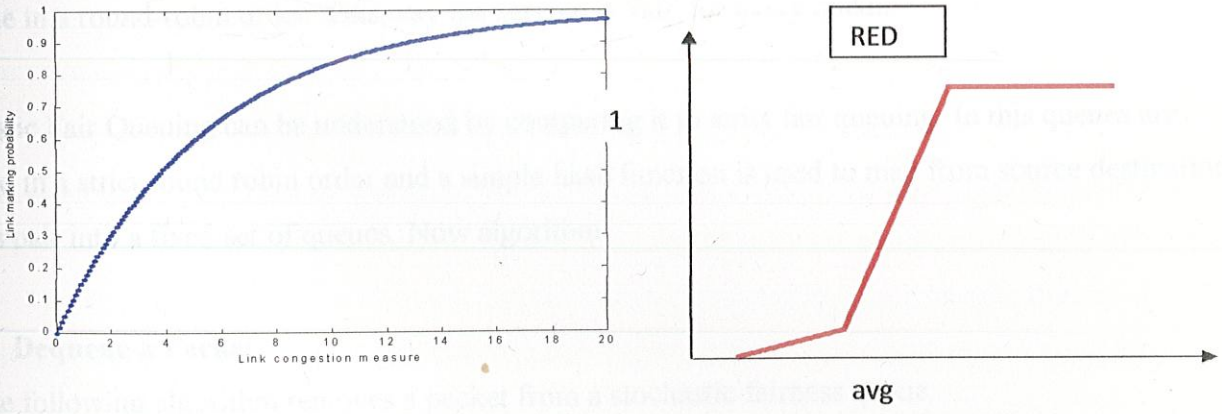


Fig. 9 Comparison of RED and REM

3.1.2.6 SFQ (Stochastic Fair Queuing)

Fair queuing (FQ) was proposed by John Nagle in 1987. FQ is the foundation for a class of queue scheduling disciplines that are designed to ensure that each flow has fair access to network resources and

to prevent a bursty flow from consuming more than its fair share of output port bandwidth. In FQ, packets are first classified into flows by the system and then assigned to a queue that is specifically dedicated to that flow. Queues are then serviced one packet at a time in roundrobin order. Empty queues are skipped. FQ is also referred to as perflow or flowbased queuing.

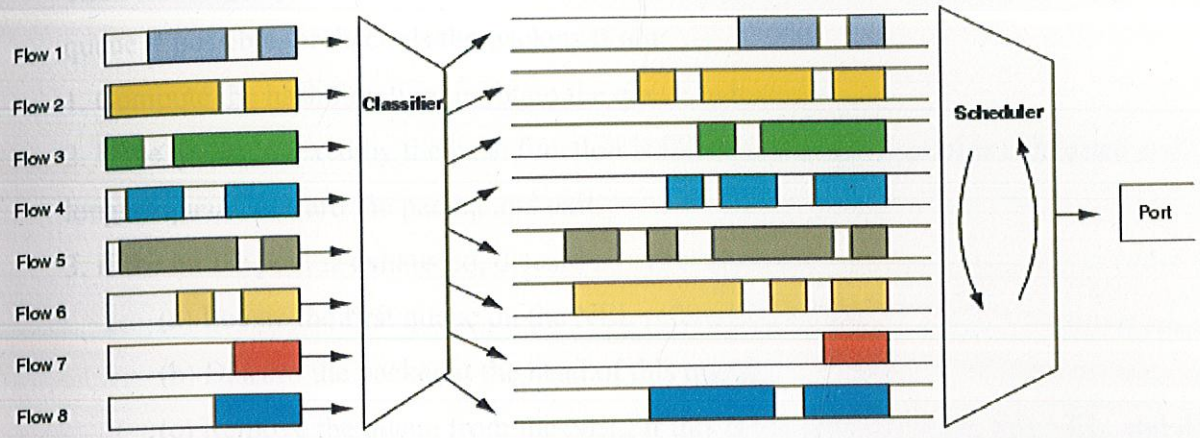


Fig.10 SQM scheme

FQ is like having several doors. When a packet arrives it is classified by the classifier and assigned to one of the doors. The door is the entry to a queue that is served together with some other, one packet at a time in a round-robin order. This way the service is 'fair' for every queue.

Stochastic Fair Queuing can be understood by comparing it to strict fair queuing. In this queues are serviced in a strict round robin order and a simple hash function is used to map from source destination address pair into a fixed set of queues. Now algorithm

1. **Dequeue a Packet**

The following algorithm removes a packet from a stochastic fairness queue:

1. If currently switching to a new perturbation of the hash function and the active list is empty, complete the switch (start outputting from the new queue).
2. If the active list is empty, exit (this implies that the entire SFQ is empty).
3. Output a packet from the queue pointed to by the roundrobin pointer.
4. Advance the round-robin pointer.

5. Delete the queue from the NEL it was in. If this NEL is now empty and the maximum-size pointer points to this NEL, decrement the maximum-size pointer

2. Enqueue a Packet

The following algorithm adds a packet to a stochastic fairness queue if possible, or discards the packets if not:

1. Compute the hash function to obtain the queue index.
2. If the queue indexed by the hash function is full or if the buffer pool is exhausted and this is the longest queue, discard the packet and exit.
3. If the buffer pool is exhausted, discard a packet from the
   (a) Locate the first queue on the NEL referenced by the
   (b) Discard the packet at the head of this queue.
   (c) Remove the queue from the NEL, if this is the only queue on this NEL, and decrement the maximum-size pointer.

3.1.2.7 SRED (Stabiized Random Early Detection)

Like RED (Random Early Detection) SRED pre-emptively discards packets with a load-dependent probability when a buffer in a router in the Internet or an Intranet seems congested. SRED has an additional feature that over a wide range of load levels helps it stabilize its buffer occupation at a level independent of the number of active connections. SRED does this by estimating the number of active connections or flows. This estimate is obtained without collecting or analyzing state information on individual flows. The same mechanism can be used to identify flows that may be misbehaving, i.e. are taking more than their fair share of bandwidth. Misbehaving flows can be identified without keeping per-flow state. Drop probabilities are adjusted according to number of active flows at the router. There is no computation of average queue length in this scheme. It assumes that TCP flows of active flows are $\propto$ number of different flows in the buffer. A misbehaving flow has a lot of packets in the buffer. When a packet arrives, compare it with a packet arrived before. If they belong to the same flow, a *hit* occurs. A list of M recently seen flows, ***zombies.***

- Information for each zombie:
  - *Count*: number of packets of this zombie received

- *timestamp*: arrival time of the most recently received packet

- Zombie list is not full
  - Insert the flow with *count* = *0*, *timestamp* = t_a
- Zombie list is full
  - Randomly pick a zombie
    - Hit: *count* += *1*, *timestamp* = t_a
    - No hit: with prob. *p* that the zombie is replaced
- The arrived packet may be dropped no matter there was a hit or not

Hit Frequency

- *P(t)* – hit frequency around the time of the t^{th} packet arrives at the buffer
- *Hit(t)* = *1* when hit; *Hit(t)* = *0*, otherwise
- $P(t) = (1 - a)P(t - 1) + a*Hit(t)$
- Proposition: $P(t)^{-1}$ is a good estimate for the effective number of active flow.

3.2 Architecture and Process Description

3.2.1 The Network Simulator - ns-2

Ns or the network simulator is a discrete event network simulator. It is popular in academia for its extensibility (due to its open source model) and plentiful online documentation. ns is popularly used in the simulation of routing and multicast protocols, among others, and is heavily used in ad-hoc networking research. ns supports an array of popular network protocols, offering simulation results for wired and wireless networks alike.

Package of NS2 includes

- Tcl
- Tk
- Otcl
- Tclcl
- Ns2
- Nam
- Xgraph (for Unix)

Tcl, Tk, Otcl, and Tclcl are support programs that are used by ns2, which is the simulation program.The directory structure is given in Fig. 11. Nam is the network simulator which provides visual views of the network simulation. Tracegraph is used to obtain statistics and produce graphical results under the Windows platform. Installation guidelines are given in the Appendix A. Various diagram explaining functioning of the Ns2

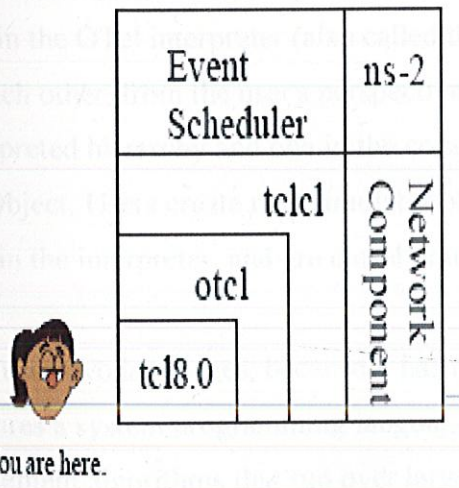


Fig 11 Architecture View of ns2

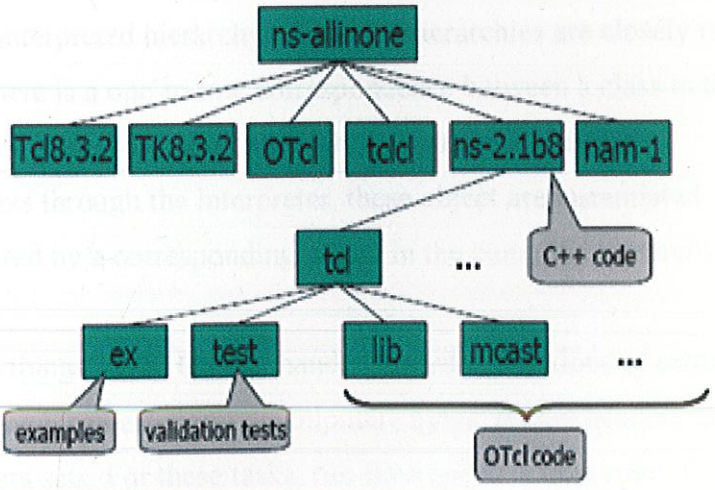


Fig. 12 ns Directory structure

The following figures shows us how ns2 help us in design network condition for simulation according to our requirement. After the designing we can see how actually the simulation will happen, the packet flow. Then we can analyse the following resultas shown in followin Fig.

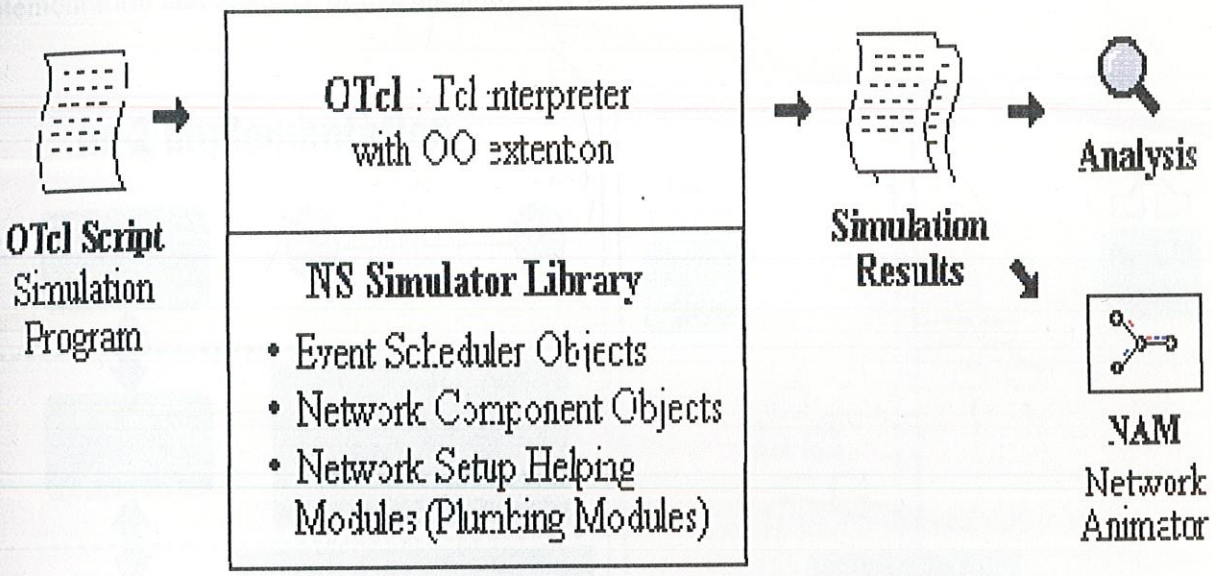


Fig 13:Simplified User's View of NS2 of output

3.2.2 Architecture Of NS2

Ns2 is an object-oriented simulator, written in C++, with an OTcl interpreter as the frontend. The simulator supports a class hierarchy in C++ (also called the compiled hierarchy), and a similar hierarchy within the OTcl interpreter (also called the interpreted hierarchy). The two hierarchies are closely related to each other, from the user's perspective, there is a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy. The root of this hierarchy is the class TclObject. Users create new simulator objects through the interpreter, these object are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy.

Ns2 uses two languages, because it has two things to do. On one hand, detailed simulations of protocols requires a system programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks, run-time speed is important. C++ is slow to change, but its speed makes it suitable for protocol implementation. On the other hand, a large part of network research involves slightly varying parameters and configurations, or exploring a number

of scenarios. In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulations), run-time of this part is less important. OTcl runs slower but can be changed very quickly making it ideal for simulation configuration. The implementation and analogy of ns2 are shown in the following diagram.

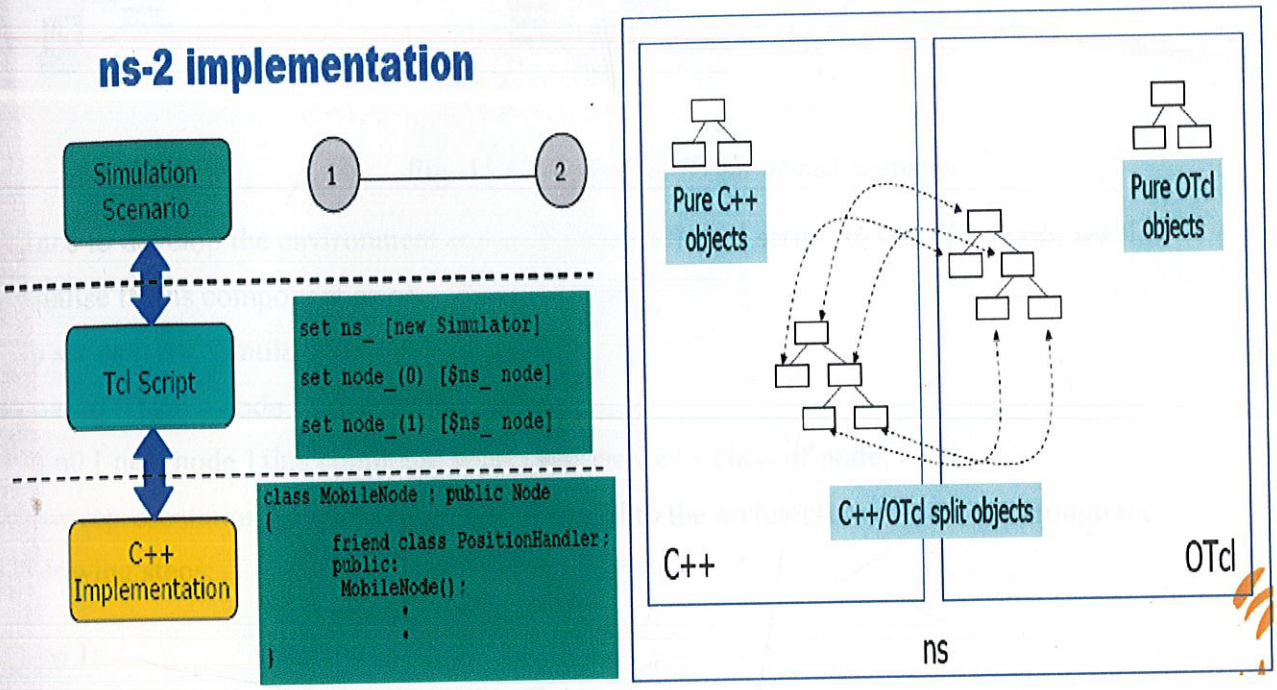


Fig 14 OTcl and C++: The Duality

The real scenario of network setup is converted into the ns2 simulation scenario. In the figure 13 an example scenario is shown in the real scenario. This situation is converted into the simulation scenario in the corresponding diagram. Each computer is treated as node in the simulation. The packets generating in a computer are transferred to destination using the routers R1 and R2. The mechanism of this transmission is deployed in the ns2 environment. Node have the property that it generates packet of desired length send it to the adjacent node using same top-down approach as in the real scenario. Since its udp flow and receiver doesn't send any information to the sender so in the simulation environment the destination node is marked with the NULL protocol as its application.

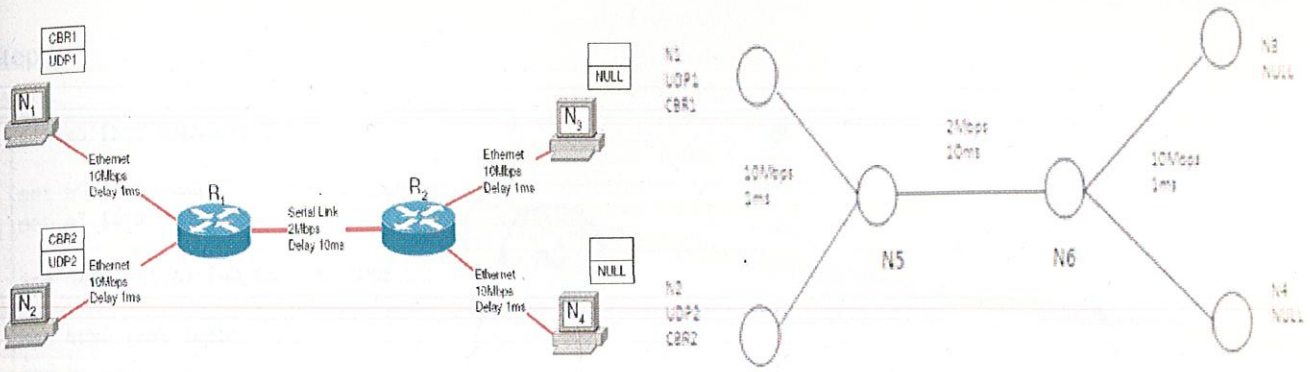


Fig. 15 Comparison of real vs ns2 scenario

Since to develop the environment we need to write the tcl script. In the TCL script we first intialise the ns component as new Simulator class.

As set ns [new Simulator]

Now to create a node we write

set n0 [new node] this command will create new c++ class of node.

Now representation how a simple code is linked to the architecture of the ns2 through the following steps:

Step 1:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"
$ns at 5.0 "stop"

$ns run
```

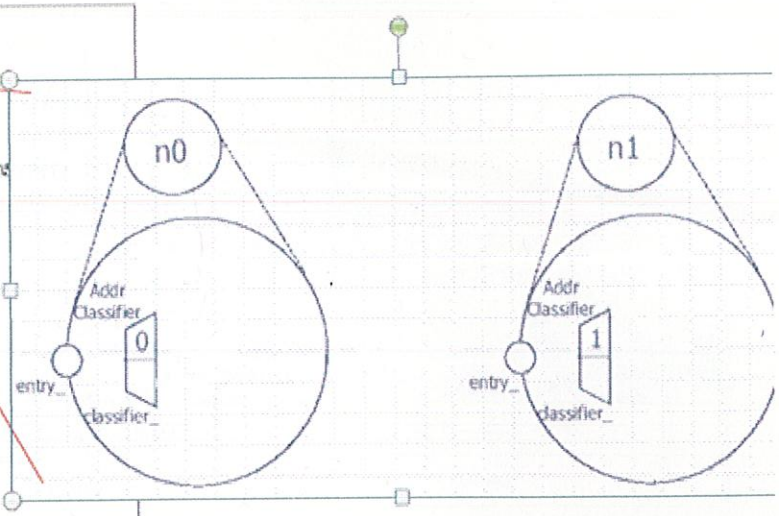
Simple.tcl

```
Simulator instproc init args {
        $self create_packetformat

        $self use-scheduler Calendar

        $self set nullAgent_ [new Age
        $self set-address-format def

        eval $self next $args
}
```

Run your program →
% ns Simple.tcl

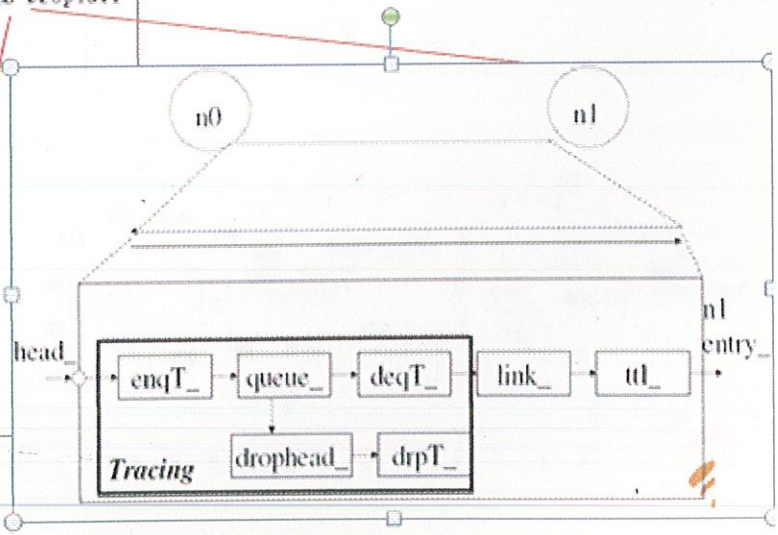
Step2:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"
$ns at 5.0 "stop"

$ns run
```



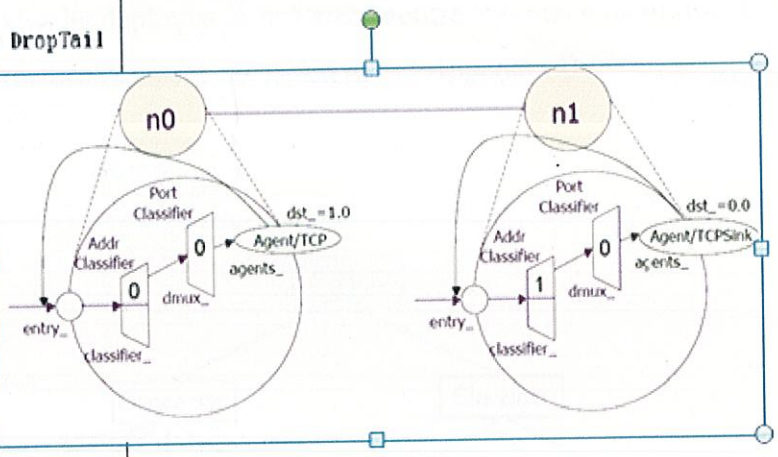
Step 3:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"
$ns at 5.0 "stop"

$ns run
```



Step 4:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"
$ns at 5.0 "stop"

$ns run
```



Step 5:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]

$ns duplex-link $n0 $n1 1Mb 10ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n1 $sink0
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns at 0.5 "$ftp0 start"
$ns at 4.5 "$ftp0 stop"
$ns at 5.0 "stop"

$ns run
```

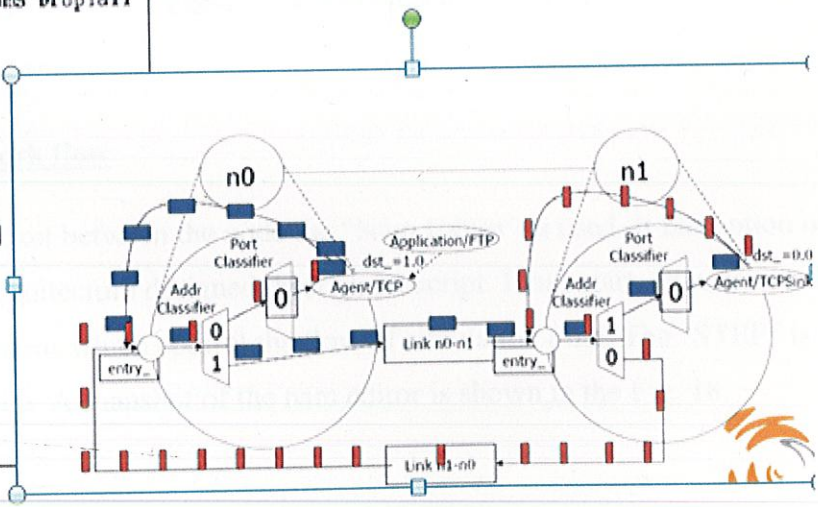


Fig 16: Steps showing how architechure of ns2 works

The complete **NS class hierarchy** is shown in Fig. 16. Now in the queue object of the hierarchy there are only two active queue management algorithm, RED and DropTail. Now the other algorithm discussed in the section of network congestion control are implement under this object only. Queue serves as the parent of all these algorithms, and they are appended as a child to this element in the hierarchy. Similarly other can also be deployed in ns2 architecture and make them run. To append the new algorithm in the ns2 architecture the steps are given in the Appendix B.

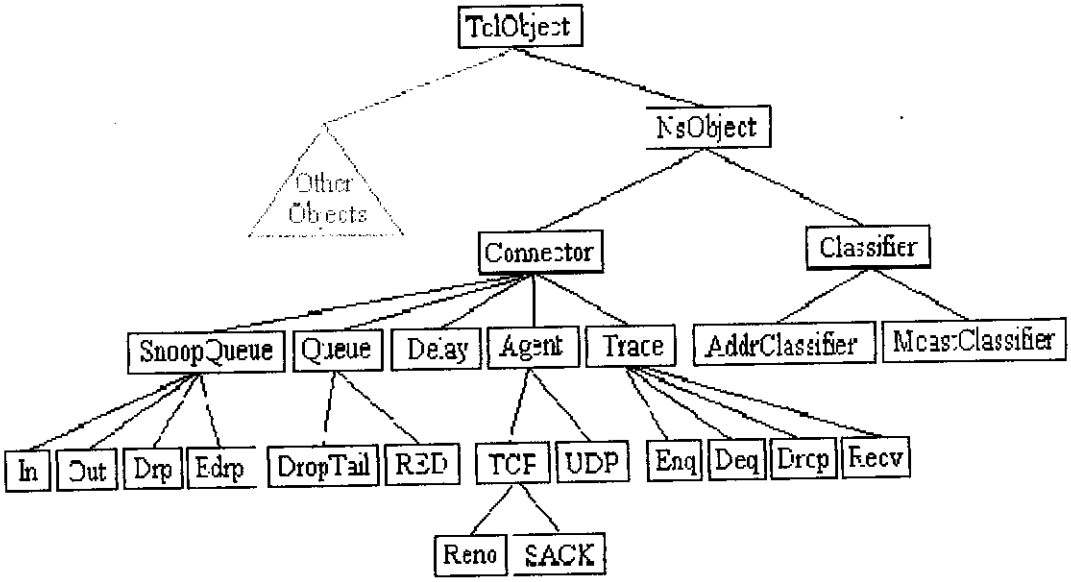


Fig. 17 Class Hierarchy

3.2.3 Visualizing the network flow

To visualize the flow of packet between the node the "Nam Editor" is used. It has option of zooming, resetting the layout of the architecture designed through tcl script. It has start, stop, forward, fast forward , rewind and fast rewind options which control the flow of simulation time. The "STEP" is used to show how fast a simulation will run. A snapshot of the nam editor is shown in the Fig. 18.

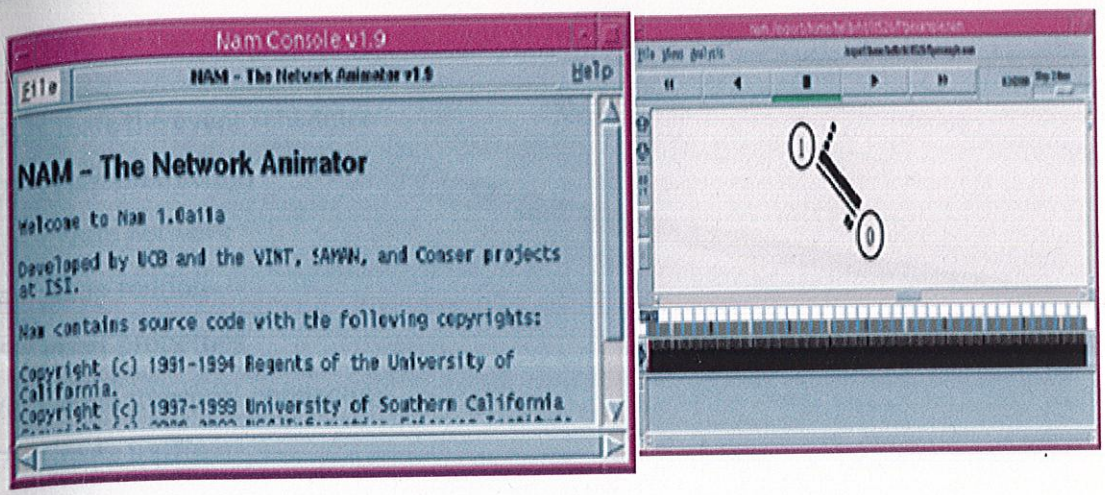


Fig. 18: Screenshot of Nam

3.4 Analysing: Trace File Format

When the ns is runed and the a trace is each event can be stored in a trace file. While tracing into an output ascii file, the trace is organized in 12 fieldsas shown in the following figure. The description of each field is shown by their name and an sample example of trace file is also shown.

event	time	from node	to node	pkt type	pkt size	flags	fid	src addr	dst addr	seq num	pkt id

```
r : receive (at to_node)                    src_addr : node.port (3.0)
+ : enqueue (at queue)                      dst_addr : node.port (3.0)
- : dequeue (at queue)
d : drop    (at queue)
```

```
r 1.3556 3 2 ack 40 ------- 1 3.0 0.0 15 201
+ 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
- 1.3556 2 0 ack 40 ------- 1 3.0 0.0 15 201
r 1.35576 0 2 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
d 1.35576 2 3 tcp 1000 ------- 1 0.0 3.0 29 199
+ 1.356 1 2 cbr 1000 ------- 2 1.0 3.1 157 207
  1.356 1 2 cbr 1000          2 1.0 3.1 157 207
```

Fig. 19 Trace file structure

Thus we can summarize the element of NS2 as:

1) Create the event scheduler
2) [Turn on tracing]
3) Create network
4) Setup routing
5) Insert properties
6) Create transport connection
7) Create traffic

3.5 AWK coding

AWK is designed for processing text-based data, either in files or data streams. The awk utility allow us to do simple operation on data files such as averaging the values of given column, summing or multiplying term by term between several columns, all data reformatting task etc.

A typical AWK program consists of a series of lines, each of them is on the form

/pattern/ { action }

Pattern is a regular expression

Action is a command.

- Most implementations of AWK use extended regular expressions by default.
- AWK looks through the input file; when it finds a line that matches pattern, it executes the command(s) specified in action.

Other line forms:

BEGIN { action }

Executes action commands at the beginning of the script execution,

END { action }

Executes action commands after the end of input.

/pattern/

Prints any lines matching pattern.

{ action }

Executes action for each line in the input.

The following example is for averaging the value

BEGIN { n1=0} {n1++}{ s=s+$4} END { print "average" s/n1}

How to run AWK file?

awk –f file1.awk file2.txt

file1.awk : is a command file

file2.txt : is a primary input file

out.txt : is an output file

3.6 GnuPlot

Gnuplot is a command-driven interactive function and data plotting program.

Gnuplot supports many types of plots in either **2D** and **3D**. It can draw using lines, points, boxes, contours, vector fields, surfaces, and various associated text. It also supports various specialized plot types.

For e.g. plot a sin(x)/x graph

Type gnuplot in your terminal on linux.

Then type plot sin(x)/x . The following graph will be generated.

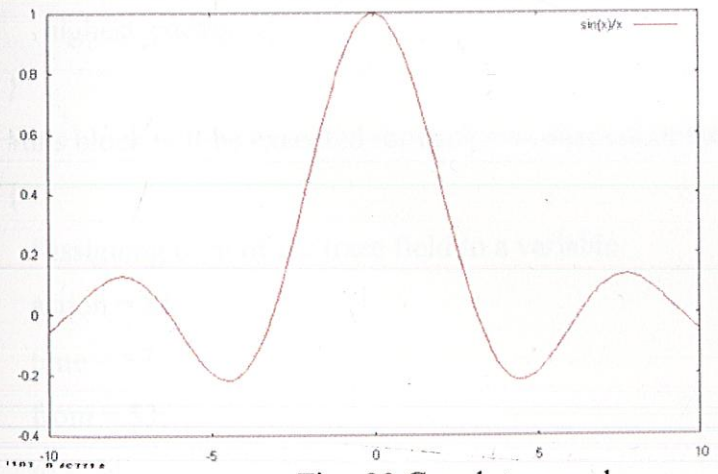


Fig . 20 Gnuplot example

Chapter 4: Performance metrics

4.1 Delay

Delay is the time elapsed while a packet travels from one point (e.g., source premise or network ingress) to another (e.g., destination premise or network degress). Delay is important because
(1) some applications (e.g., audio and video applications) do not perform well (or at all) if end-to-end delay between nodes is large relative to some threshold value,
(2) erratic variation in delay makes it difficult (or impossible) to support many real-time applications,
(3) the larger the value of delay, the more difficult it is for transport layer protocols to maintain high bandwidths,
(4) the minimum value of this characteristic indicates the delay due only to propagation and transmission delay, which will likely be experienced when the path traversed is lightly loaded, and
(5) values of this characteristic above the minimum means the congestion present in the path. This characteristic can be specified in a number of different ways, including average delay, variance of delay (jitter), and delay bound.

IT's AWK code with respect to the trace file:

```
BEGIN {
    highest_packet_id = 0;
}
#this block will be executed for each row element of the trace file
{
  #assigning each of the trace field to a variable
  action = $1;
  time = $2;
  from = $3;
  to = $4;
  type = $5;
  pktsize = $6;
  flow_id = $8;
```

```
src = $9;
dst = $10;
seq_no = $11;
packet_id = $12;

if ( packet_id > highest_packet_id )
    highest_packet_id = packet_id;

if ( start_time[packet_id] == 0 )
    start_time[packet_id] = time;

#now assiging each packet received time if it reach to the receiver end and it is not dropped
   if ( action == "r" && (to==dst)) {
     end_time[packet_id] = time;
    } else {
   end_time[packet_id] = -1;
  }
}
#this part of code will execute at the completion of the row field in the trace file. It is writing the delay
time for each packet into output.
END {
  for ( packet_id = 0; packet_id <= highest_packet_id; packet_id++ ) {
    start = start_time[packet_id];
    end = end_time[packet_id];
    packet_duration = end - start;

    if ( start < end ) printf("%f    %f\n",start, packet_duration);
  }
}
```

4.2 Packet Loss

Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss during the steady state is another important property of a congestion control scheme. Loss is important because

(1) some applications (e.g., audio and video applications) do not perform well (or at all) if end-to-end loss between nodes is large relative to some threshold value,

(2) excessive packet loss may make it difficult to support certain realtime applications,

(3) the larger the value of packet loss, the more difficult it is for transport-layer protocols to maintain high bandwidths,

(4) the sensitivity of real-time applications and of transport-layer protocols to loss become especially important when very large delay-bandwidth products must be supported, and

(5) the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself. This characteristic can be specified in a number of different ways, including loss rate, loss patterns, loss free seconds, and conditional loss probability.

IT's AWK code with respect to the trace file:

```
BEGIN {
# Initialization. Set two variables. fsDrops: packets drop. numFs: packets sent
      fsDrops = 0;
      numFs = 0;
      ratio = 0;
}


{
  action = $1;
  time = $2;
  from = $3;
  to = $4;
  type = $5;
  pktsize = $6;
```

```
 flow_id = $8;
 src = $9;
 dst = $10;
 seq_no = $11;
 packet_id = $12;
#here the 20 and 21 node act as a router where the packet is lodt due to the algorithm established on it.
 if ((from==20 || from==21) && (to==21 || to==20) && (type=="cbr" || type=="tcp") && action ==
"+")
      numFs++;
 if (action == "d")
      fsDrops++;
}
END {
     ratio = numFs/fsDrops;
     printf("number of packets sent:%d lost:%d  ratio:%d\n", numFs, fsDrops,ratio);
}
```

4.3 Throughput

This is the main performance measure characteristic, and most widely used. This measures how soon the receiver is able to get a certain amount of data send but the sender. It is determined as the ratio of the total data received by the end to the connection time. Throughput is an important factor which directly impacts the network performance.

IT's AWK code with respect to the trace file:

```
#measure_interval will give the rate after every 0.5 seconds
BEGIN {
   measure_interval = 0.5;
   bits = 0;
   first_time = 0;
 }
```

```
{
  action = $1;
  time = $2;
  from = $3;
  to = $4;
  type = $5;
  pktsize = $6;
  flow_id = $8;
  src = $9;
  dst = $10;
  seq_no = $11;
  packet_id = $12;
# measure the rate of transformation of packet
  if ((action == "r") && ((from == 20 && to ==21) || ((from==21) && (to==20)))) {
      if ((time - first_time) > measure_interval) {
        first_time = first_time + measure_interval;
        rate = (bits/1000000)/first_time;
        printf("%f %f\n", first_time, rate);
       }
      bits = bits + $6 * 8;
    }
}
END {
measure_interval = 0.5;
first_time = first_time + measure_interval;
rate = (bits/1000000)/first_time;
printf("%f %f\n", first_time, rate);
}
```

4.4 Queue Length

A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving the system after being served. In most cases, four basic characteristics of queuing processes provide an adequate description of a queuing system in networks:
(1) arrival pattern of packets,
(2) service pattern of schedulers,
(3) queue discipline, and
(4) system capacity.
Thus queue length is very important characteristic to determine that how well the active queue management of the congestion control algorithm is been working.

IT's AWK code with respect to the trace file:

```
BEGIN {

   measure_interval = 0.5;
   qlength = 0;
   first_time = 0;
}

{
  action = $1;
  time = $2;
  from = $3;
  to = $4;
  type = $5;
  pktsize = $6;
  flow_id = $8;
  src = $9;
  dst = $10;
  seq_no = $11;
  packet_id = $12;
#this condition is checking the combined queue length oat the router
```

```
if ((action == "+") && (from == 20)) {
    qlength = qlength + 1;
}
if ((action == "-") && (from == 20)) {
     qlength = qlength -1;
}
if ((action == "+") && (from == 21 )) {
    qlength = qlength + 1;
}
if ((action == "-") && (from == 21)) {
     qlength = qlength -1;
}
if(((action == "+") || (action == "-"))&&((from == 20) || (from ==21)) ){
   if ((time - first_time) > measure_interval) {
     first_time = first_time + measure_interval;
     printf("%f %f\n", first_time, qlength);
    }
 }
}
END {
measure_interval = 0.5;
first_time = first_time + measure_interval;
printf("%f %f\n", first_time, qlength);
}
```

4.5 Jitter

Packet jitter is the variability of packet delays within the same packet stream. Packet jitter, if not suppressed, can easily lead to unintelligible audio and video One of the components of end-to-end delay is the random queuing delays in the routers. Because of these random queuing delays within the

network, the time from when a packet is generated at the source until it is received at the receiver can fluctuate from packet to packet. This phenomenon is called **jitter**.

IT's AWK code with respect to the trace file:

```
#This program is used to calculate the jitters for CBR

# jitter =((recvtime(j)-sendtime(j))-(recvtime(i)-sendtime(i)))/(j-i),  j > I where j and I are the packet seruence number

BEGIN {   # Initialization

    highest_packet_id = 0;

}

{

  action = $1;    time = $2;

  from = $3;    to = $4;

  type = $5;    pktsize = $6;

  flow_id = $8;    src = $9;

  dst = $10;

  seq_no = $11;

  packet_id = $12;

  if ( packet_id > highest_packet_id ) {

        highest_packet_id = packet_id;

    }
```

```
#Record the transmission time

if ( start_time[packet_id] == 0 )  {

        # Record the sequence number

        pkt_seqno[packet_id] = seq_no;

        start_time[packet_id] = time;

  }

#Record the receiving time for flow_id=2

  if ( flow_id == 2 && action != "d" ) {

    if ( action == "r" ) {

          end_time[packet_id] = time;

    }

   } else {

    end_time[packet_id] = -1;

  }

}

#entering the jitter time for each sequence number

END {

      last_seqno = 0;

      last_delay = 0;
```

```
seqno_diff = 0;

for ( packet_id = 0; packet_id <= highest_packet_id; packet_id++ ) {

        start = start_time[packet_id];

        end = end_time[packet_id];

         packet_duration = end - start;

         if ( start < end ) {

                seqno_diff = pkt_seqno[packet_id] - last_seqno;

                delay_diff = packet_duration - last_delay;

                if (seqno_diff == 0) {

                  jitter =0;

          } else {

                  jitter = delay_diff/seqno_diff;

        }

        printf("%f %f\n", start, jitter);

        last_seqno = pkt_seqno[packet_id];

        last_delay = packet_duration;

   }

}}
```

Chapter 5 Simulation in NS2

5.1 Basic Comparison of RED and DropTail

DropTail and RED are the only algorithms which are implemented on the routers at present. These algorithms affect the performance of a network, when there is a bottoleneck link present in the system. As Red uses the congestion avoidance and congestion control, so obviously it is more effective. But in the case when the burstiness of the traffic has very low chance of happening, then RED will unnecessary drop the packet which decreases the throughput.

5.1.1 Simulation Scenario

The simulation setting consists of two nodes transmitting traffic, one generating the TCP traffic and other UDP. They both are passing through the same router n1 and n2, which act as bottleneck link to the system. At the other end of the bottleneck link there are two nodes, which act as receiver to traffic generated at the other end of the traffic.

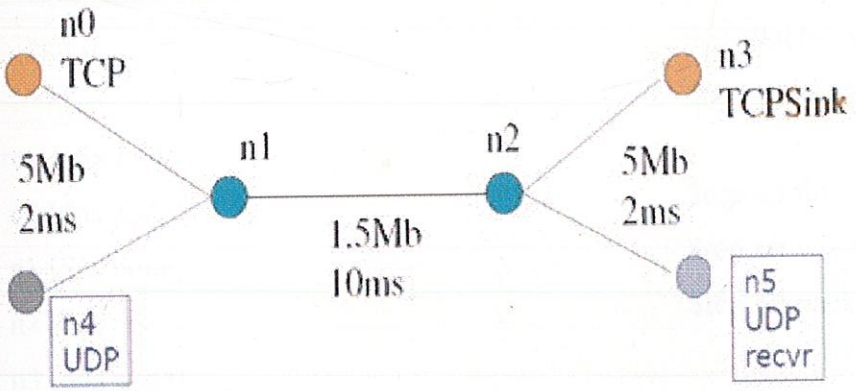


Fig. 21 Simulation scenario 1

5.1.2 Code

```
set ns [new Simulator]
#Define different colors for dataflows (for NAM)
$ns color 1 Blue
```

```
$ns color 2 Red

#Open the Trace files
set file1 [open out.tr w]
$ns trace-all $file1

#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2

#Define a 'finish' procedure
proc finish {} {
    global ns file1 file2
    $ns flush-trace
    close $file1
    close $file2
    exec nam out.nam &
    exit 0
}
#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 100ms RED
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 20

#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink

$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#Setup a UDP connection
set udp [new Agent/UDP]
```

```
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2
#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"
$ns at 124.5 "$cbr stop"
$ns at 125.0 "finish"
$ns run
```

5.1.3 Performance Metrics

The two are compared on basis of three performance metrics which are shown below:

a) Delay diagram

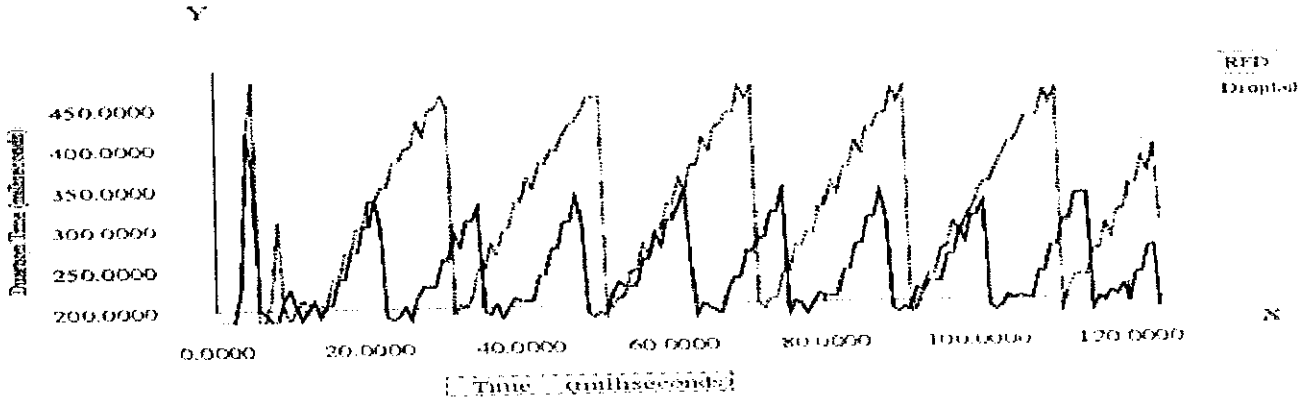


Fig. 22 Delay for RED and DropTail

b) Throughput Diagram

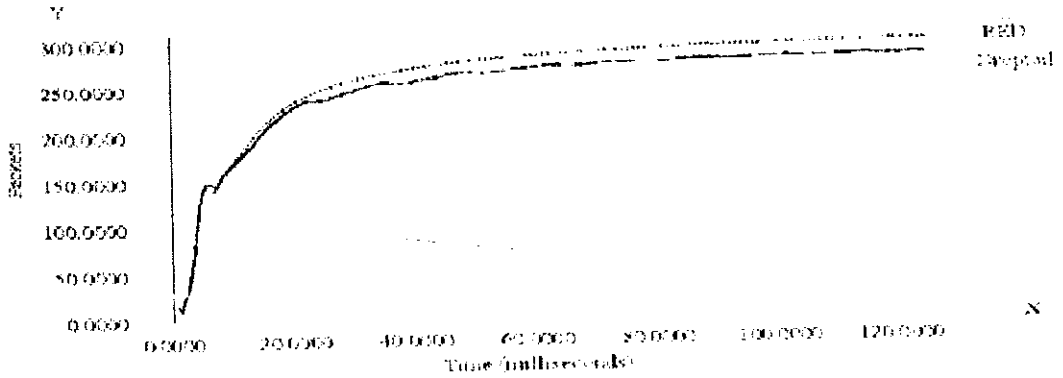


Fig. 23 Throughput for RED and DropTail

c) Jitter diagram

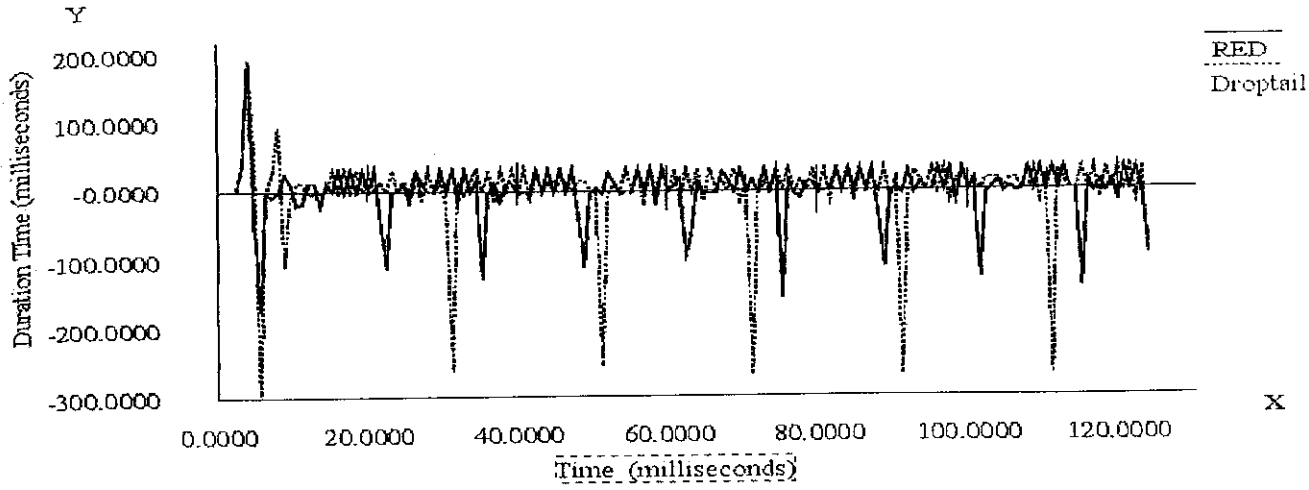


Fig. 24 Jitter for RED and DropTail

5.2 Comparing various congestion control algorithm

The algorithms compared here are first deployed into the ns2 architecture. All these are still theoretical concept and still research are ongoing on them. The following simulation scenario is generated to compare their performance on the simulation setting as shown. The traffic generated to all them are same and the time of their sending is also same. These algorithm are taken as only their code are built for the ns2 utilization. Others can be same way compared, just building their code in ns2 way and appending in to the hierarchy.

5.2.1 Simulation Scenario

There are ten nodes at the each side of the bottleneck link. At each side five nodes are acting as a source and five are acting as a destination so that the both router are applying the congestion control algorithm. There are two way traffic in the system. There are nine tcp source and only one udp source. Router having the delay time more than the outing line, which are desperately made so that the queue should build up for the algorithms to work.

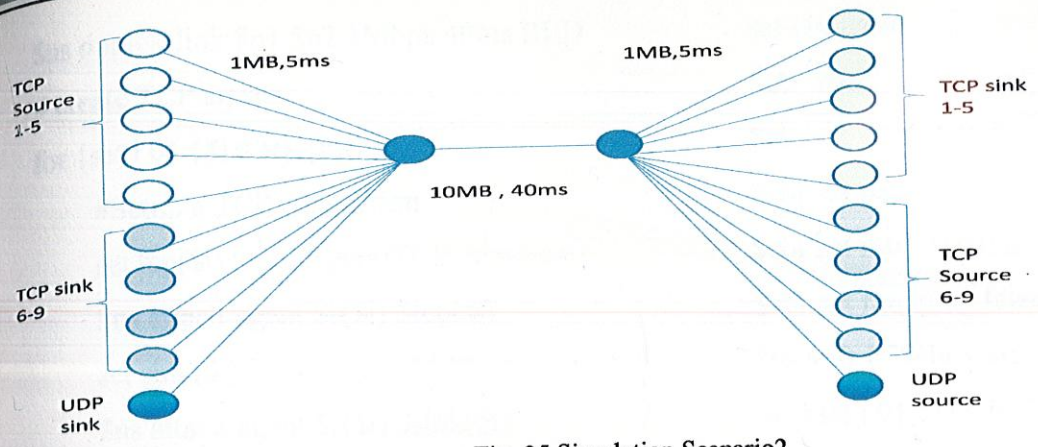


Fig. 25 Simulation Scenario2

5.2.2 Code

```
set ns [new Simulator]
$ns color 1 Blue
$ns color 2 Red
$ns color 3 White
$ns color 4 Brown
$ns color 5 Black
$ns color 6 Purple
$ns color 7 Orange
$ns color 8 Yellow
$ns color 9 Pink
$ns color 10 Green
#Open the Trace files
set file1 [open out.tr w]
$ns trace-all $file1
#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2
#set the no of TCP flows here
#set nodenum 10
set start_time 0.0
set finish_time 125.0
# create the nodes
for {set i 0} {$i < 10} {incr i} {
   set s($i) [$ns node]
   set r($i) [$ns node]
   }
set n1 [$ns node]
set n2 [$ns node]
# create the links between the senders and n1,
receivers and n2
for {set i 0} {$i < 5} {incr i} {
   $ns duplex-link $s($i) $n1 10Mb 5ms
DropTail
   $ns duplex-link $r($i) $n2 10Mb 5ms
DropTail
   }
for {set i 5} {$i < 10} {incr i} {
   $ns duplex-link $r($i) $n1 10Mb 5ms
DropTail
   $ns duplex-link $s($i) $n2 10Mb 5ms
DropTail
#Bottle neck link between between n1 and n2
```

```
$ns duplex-link $n1 $n2 1Mbps 40ms RED
# create TCP agents
for {set i 0} {$i < 9} {incr i} {
      #Setup a TCP connection
      set tcp($i) [new Agent/TCP/Newreno]
      $ns attach-agent $s($i) $tcp($i)
      set sink($i) [new Agent/TCPSink/DelAck]
      $ns attach-agent $r($i) $sink($i)
          $ns connect $tcp($i) $sink($i)
          $tcp($i) set fid_ [expr ($i + 1)]
          $tcp($i)  set window_ 8000
          $tcp($i)  set packetSize_ 552
          #Setup a FTP over TCP connection
          set ftp($i) [new Application/FTP]
          $ftp($i) attach-agent $tcp($i)
          $ftp($i) set type_ FTP

}
#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $s(9) $udp
set null [new Agent/Null]
$ns attach-agent $r(9) $null
$ns connect $udp $null
$udp set fid_ 10
#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false
$ns at 0.1 "$cbr start"
for {set j 0} {$j < 9} {incr j} {
  set k [expr $j*5]
  $ns at [expr 1.0+$k] "$ftp($j) start"
  $ns at [expr 124.0-$k] "$ftp($j) stop"
}
$ns at 124.5 "$cbr stop"
$ns at 125.0 "finish"
#Define a 'finish' procedure
proc finish {} {
      global ns file1 file2
      $ns flush-trace
      close $file1
      close $file2
      puts "running    "
      exec nam out.nam &
      exit 0
}
$ns run
```

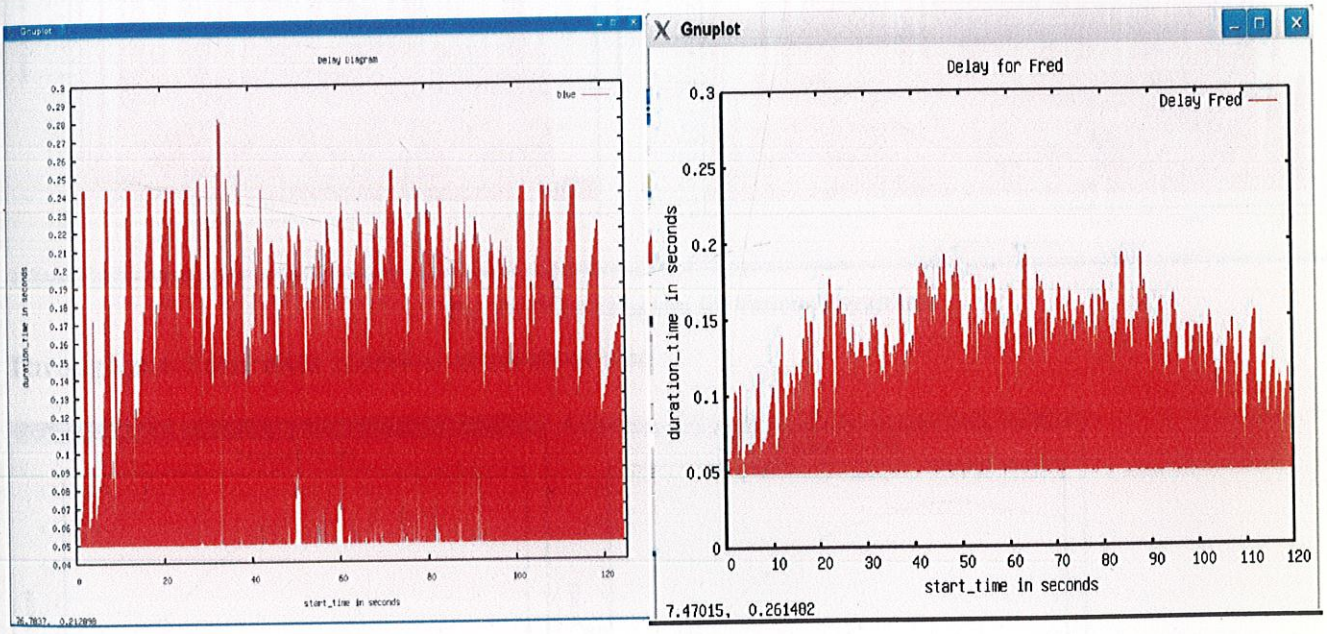
5.2.3 Performance Comparison for Various algorithms

The sample code of gnuplot to plot the graph to measure the performance

```
set terminal postscript portrait  enhanced mono lw 2 "Helvetica" 14
replot
```

```
set terminal x11
set size 1,1
set title "Througput Diagram"
set xlabel "Time in second"
set ylabel "Throughput in Mbps"
set key 3,1.8
set xr [0:5]
set yr [0:2]
plot    "thr1" using 1:2 title ' Flow1' with lines, \
        "thr2" using 1:2 title 'Flow2' with lines
# for plotting to graph in same for comparison. Similarly the single multiple can be plotted using the
above graph
```

a) Delay : Time between sending and receiving packet



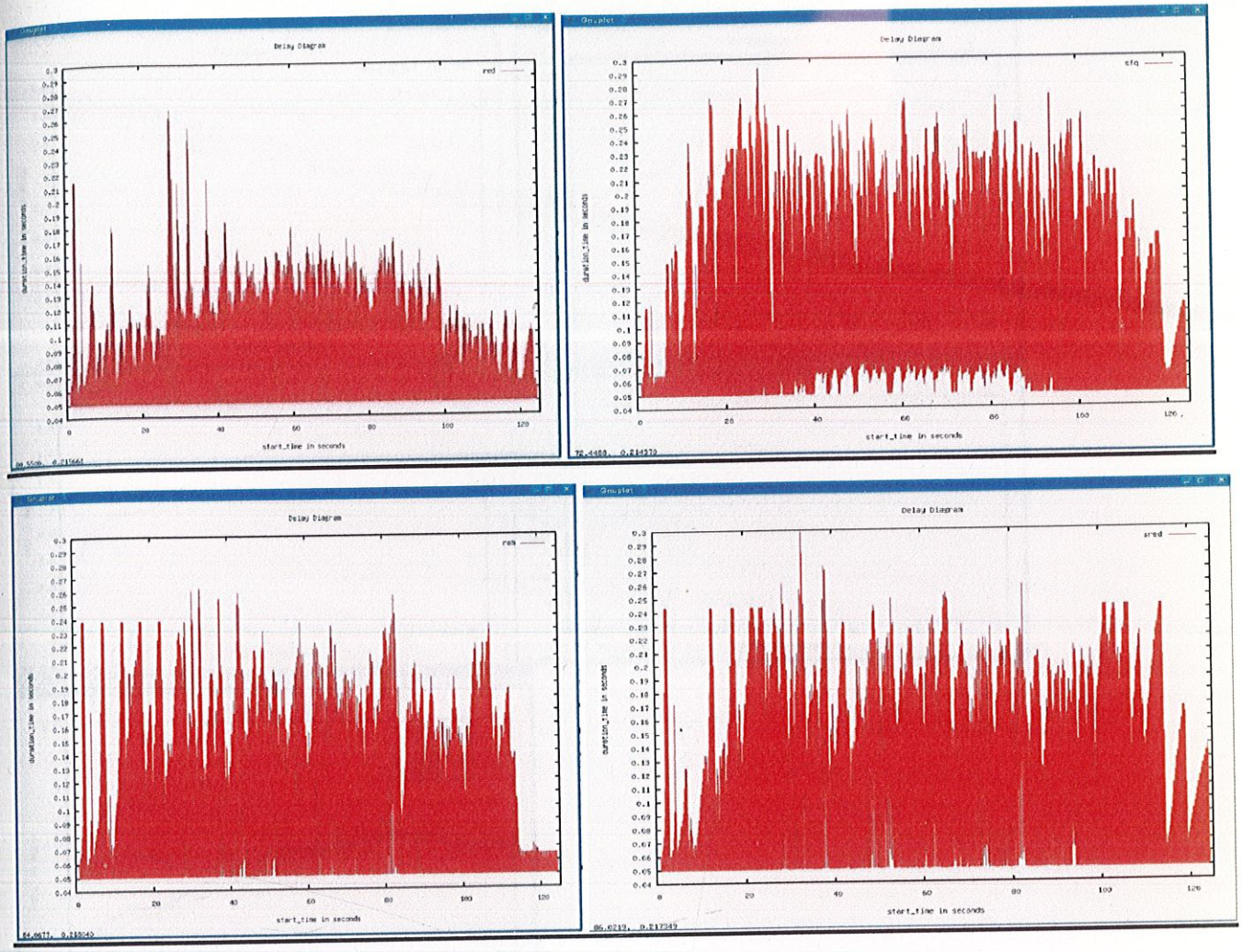
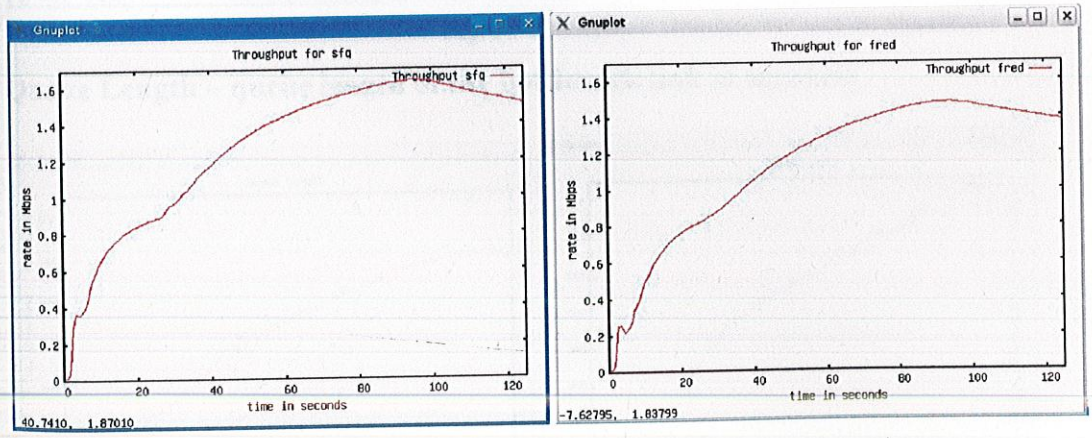


Fig. 26 Delay Diagram for various algorithms

b) Throughput-Total data received at receiver end



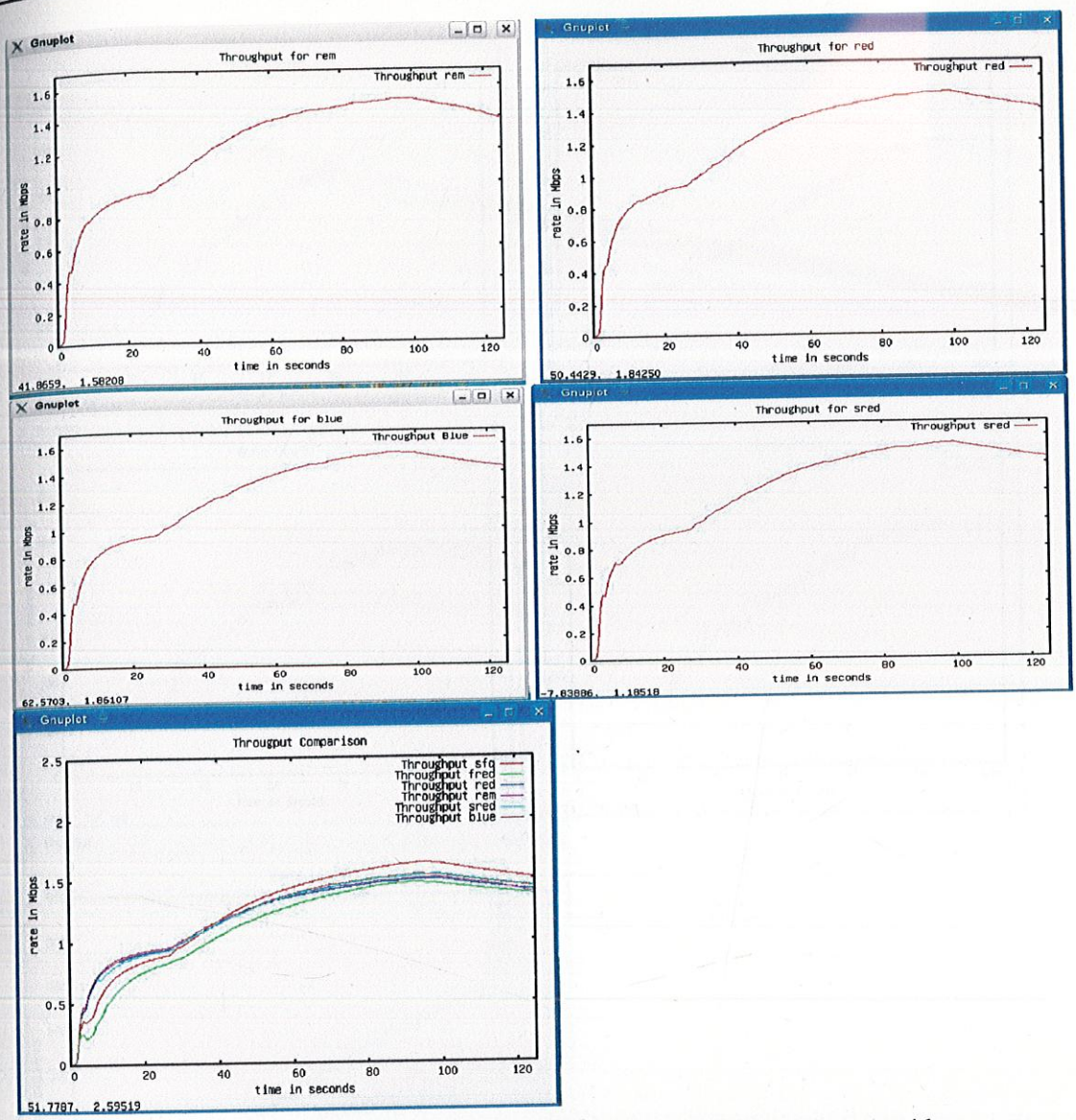
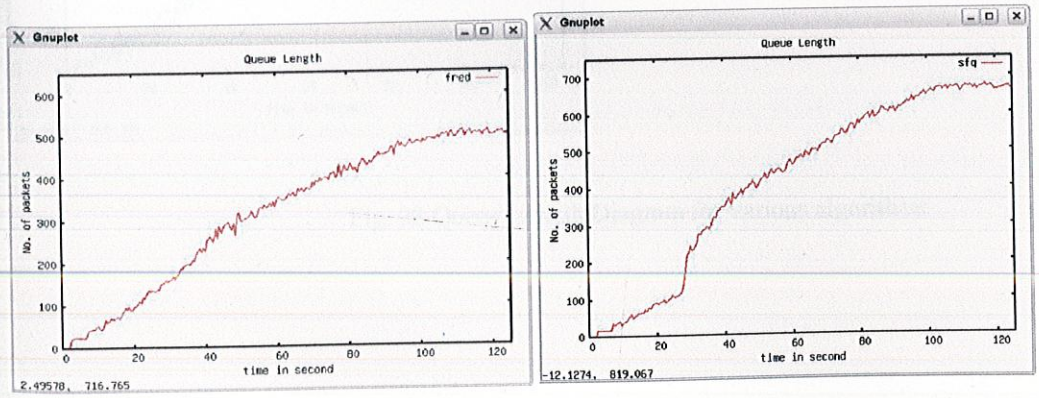


Fig. 27 Throughput Diagram for various algorithms

c) Queue Length – queue length of the bottleneck link in scenario



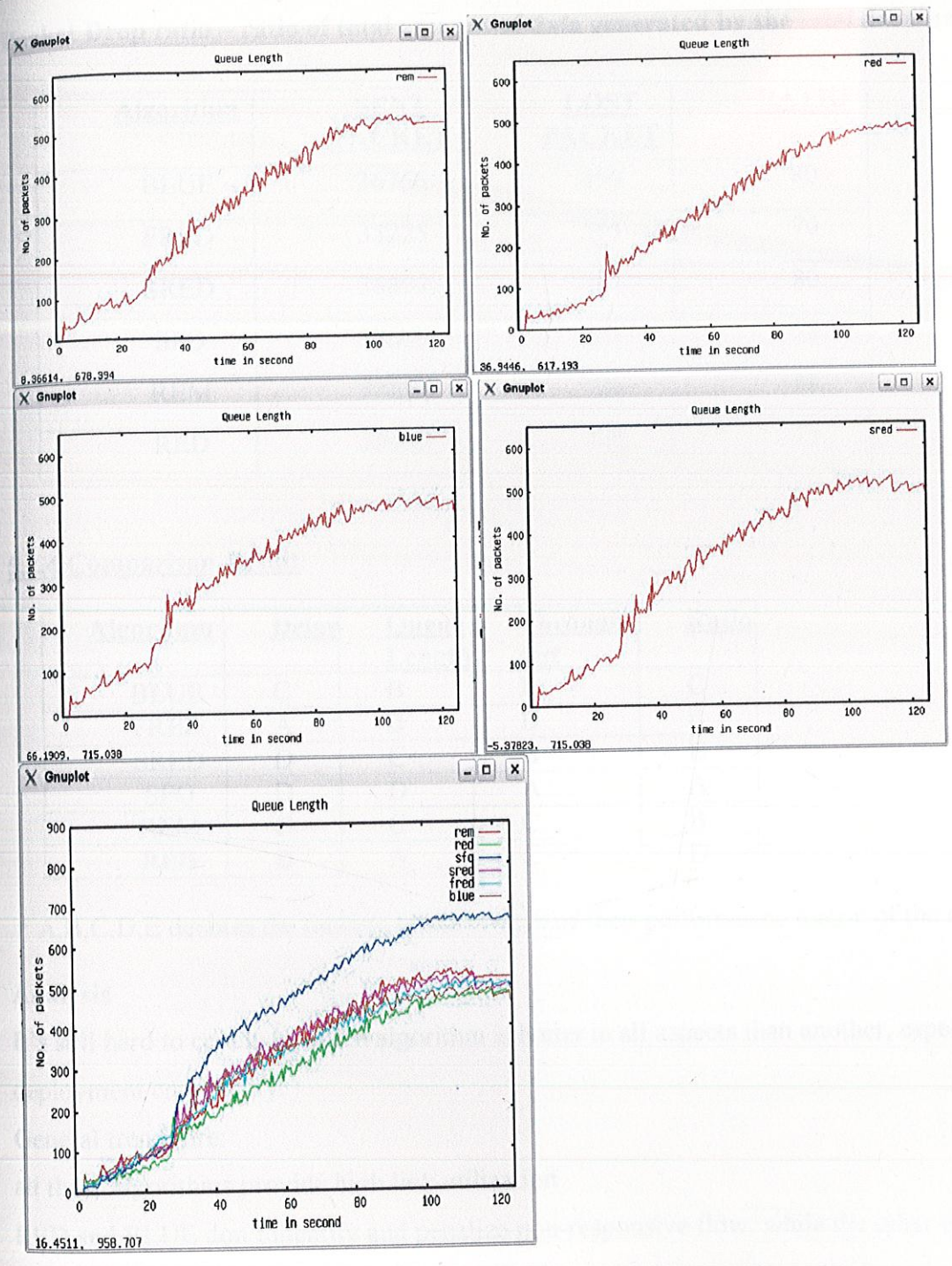


Fig. 28 Queue Length Diagram for various algorithms

d) **Packet Drop ratio – ratio of total amount of data generated by the total drop packet**

Algorithm	SENT PACKET	LOST PACKET	RATIO
BLUE	36766	459	80
FRED	35225	501	70
SRED	36897	457	80
SFQ	38689	661	58
REM	35592	521	68
RED	35949	478	75

5.2.4 Comparison Result

Algorithm	Delay	Queue Length	Through put	Ratio
BLUE	C	B	B	C
FRED	A	B	D	E
SRED	D	C	B	E
SFQ	D	D	A	A
REM	C	C	C	B
RED	B	A	C	D

* A,B,C,D,E denotes the ranking grade in each of their performance metric of the algorithm

Analysis

- It's still hard to conclude which algorithm is better in all aspects than another, especially considering the deployment complexity.
- General trends are:
  - all these algorithms provide high link utilization
  - RED and BLUE don't identify and penalize non-responsive flow, while the other three algorithms maintains fair sharing among different traffic flows

Chapter 6 Design and Development of Tool

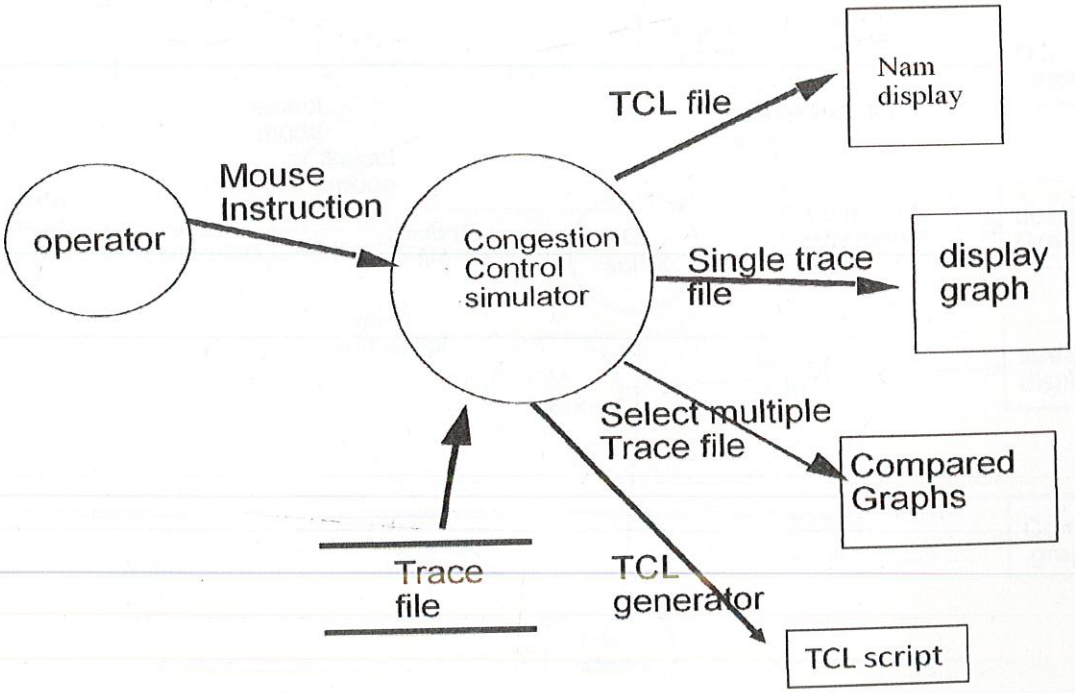
In the previous chapter we had seen the simulation of two scenarios and their comparison through the various performance metrics. Bur all the setting are assumed and they are not fixed. Since in real scenario the behavior of traffic is highly unpredictable so previous way of comparing through the ns, trace file is highly tedious task. So we have developed the tool for the comparisons of these algorithms on different conditions, which are just the few click tasks for the users. We have built a user interface tool in which the graphs are easily generated for any type scenario.

The salient features of the tools:

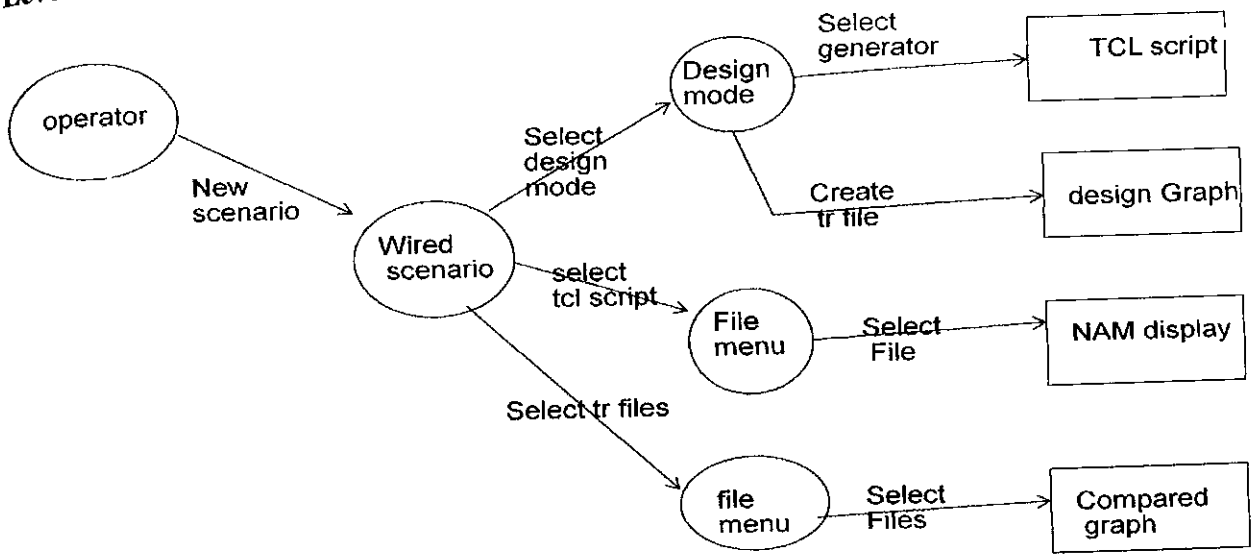
a) User interface for generating the scenario
b) Option to run the simulation in ns2 from the tool itself
c) Single graph for the performance is generated
d) Graph comparing various algorithm
e) Automated TCL script generation

6.1 Data Flow Diagram

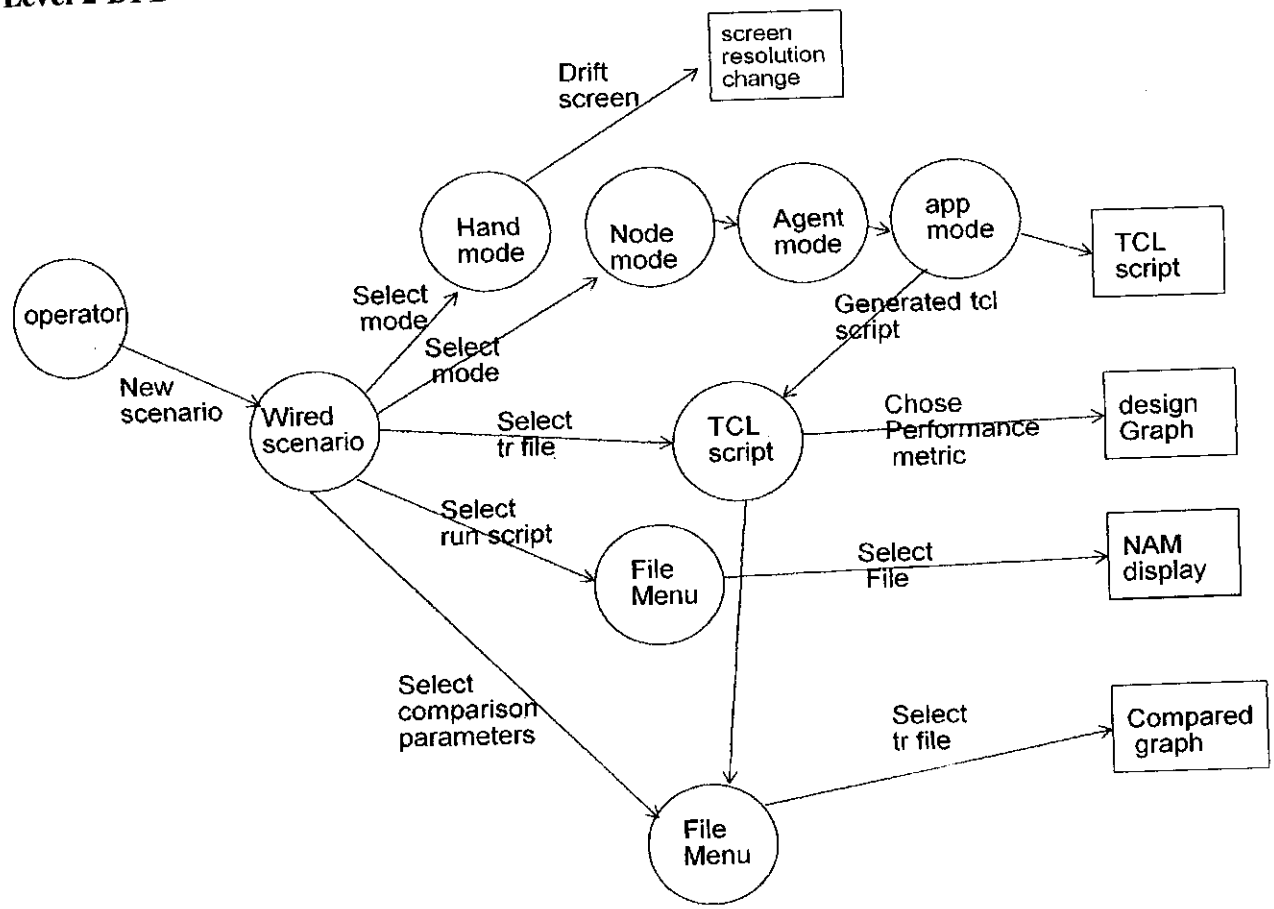
Level 0 DFD



Level 1 DFD

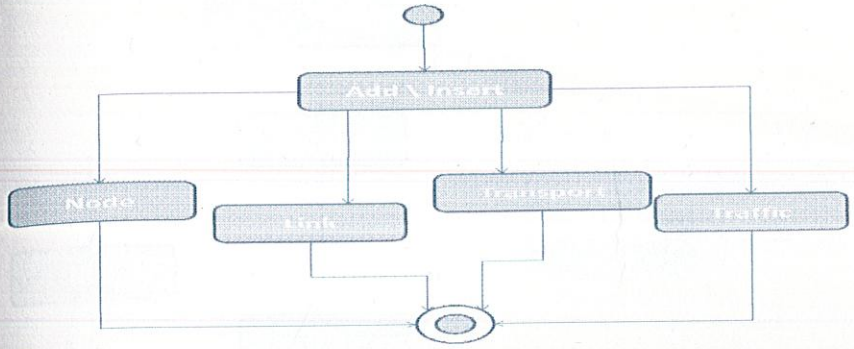


Level 2 DFD

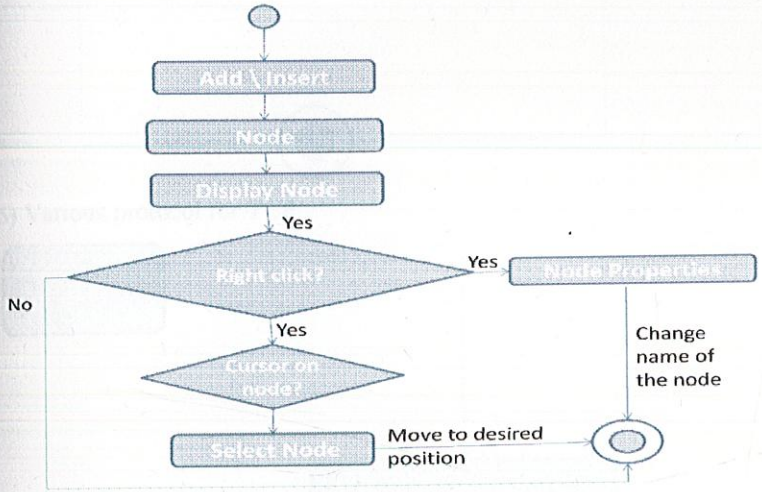


6.2 Activity Diagram

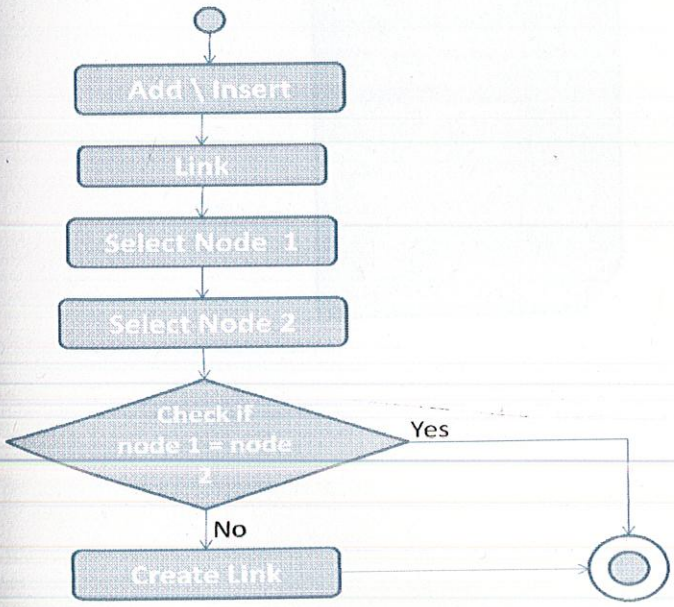
1) Basic activities



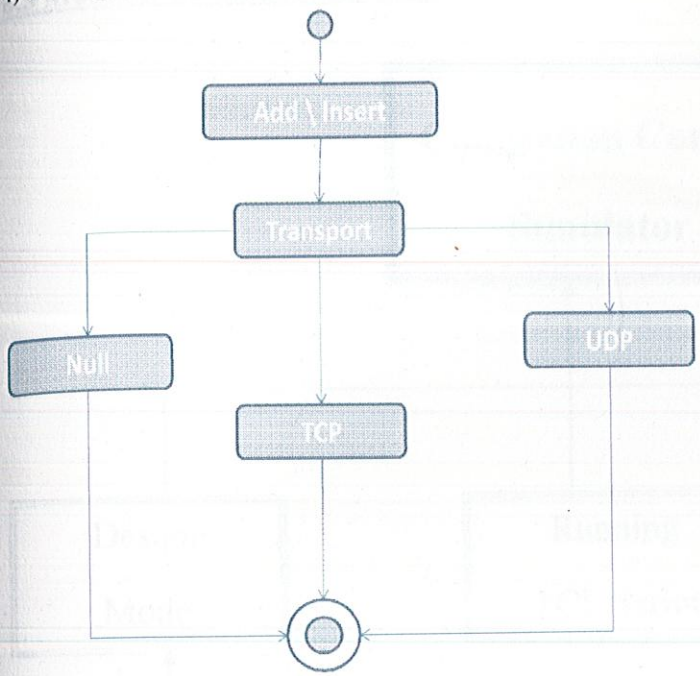
2) Activity for inserting node



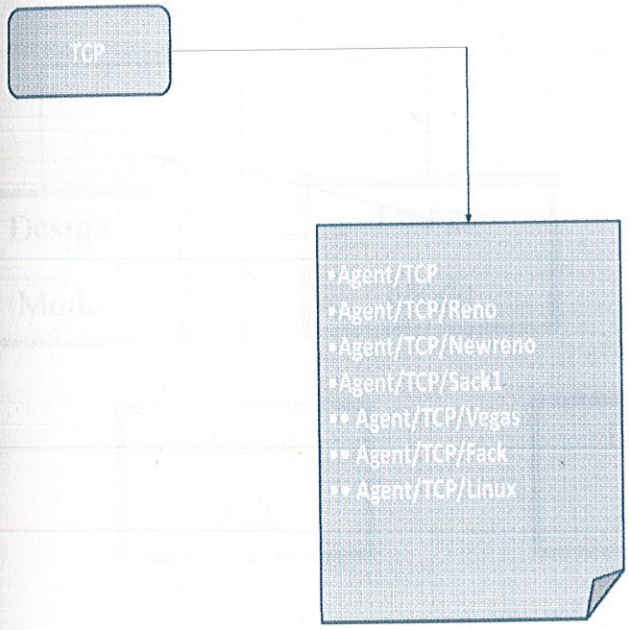
3) Activity for inserting node



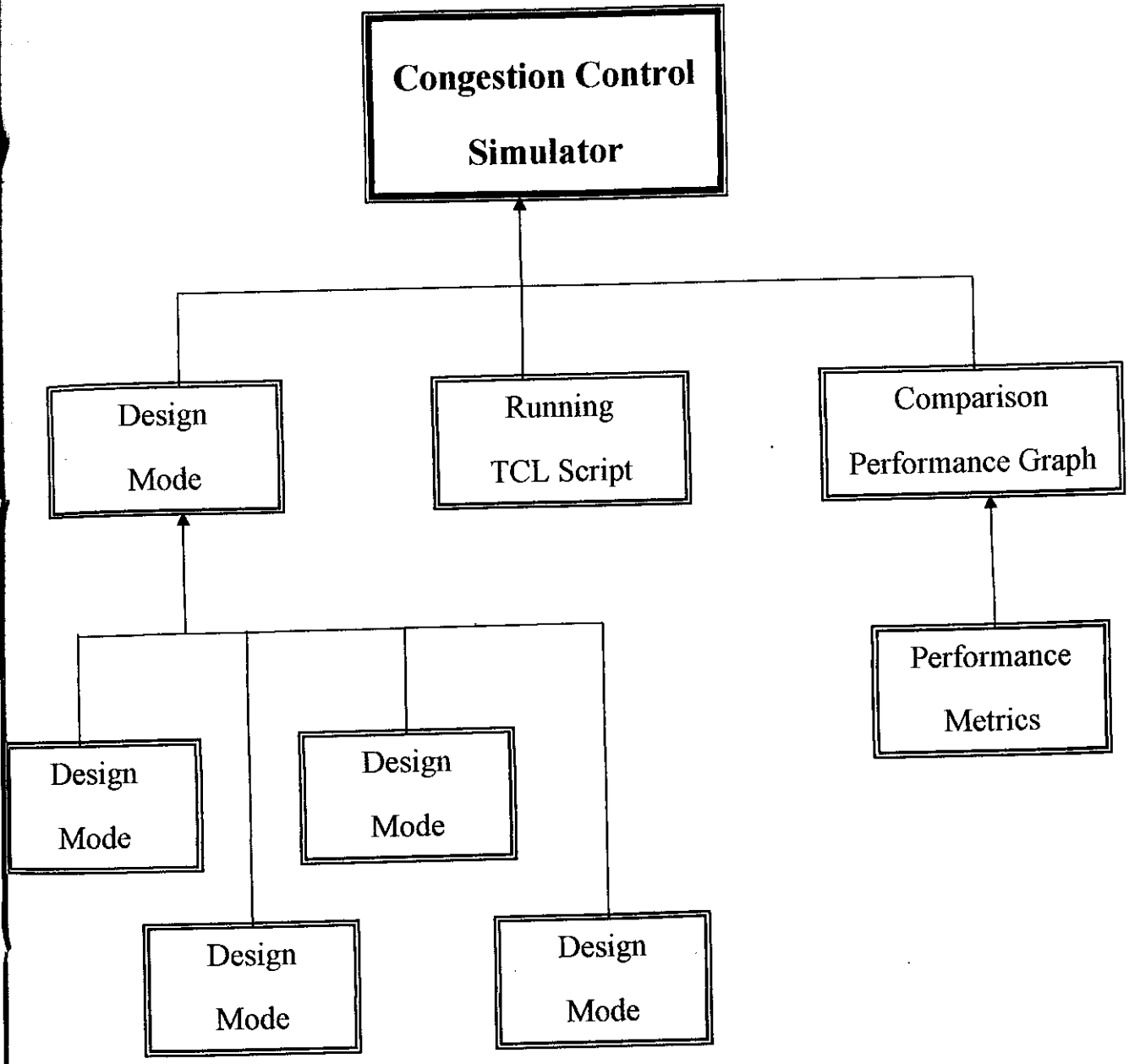
4) Activity for inserting agents



5) Various protocol for TCP

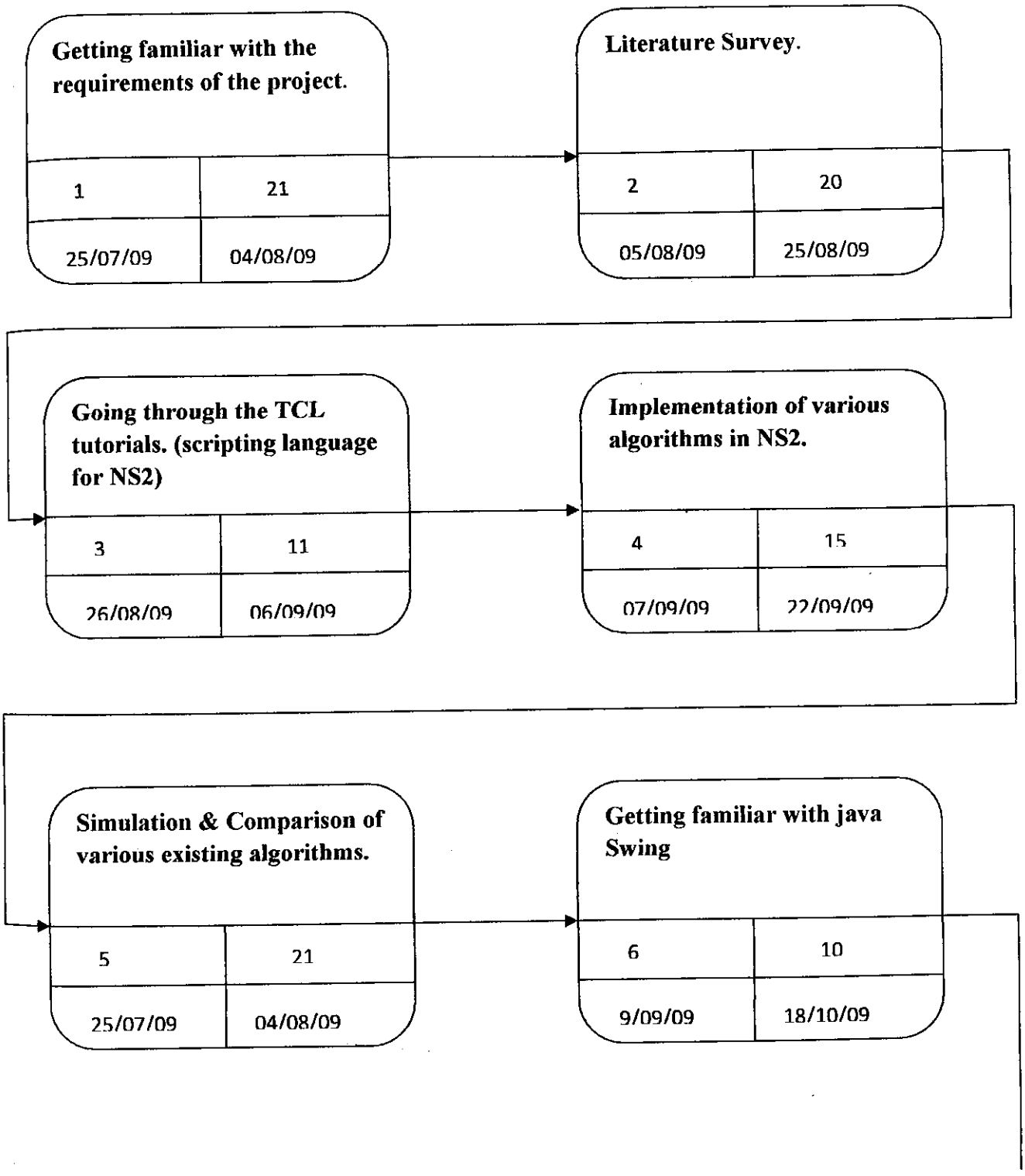


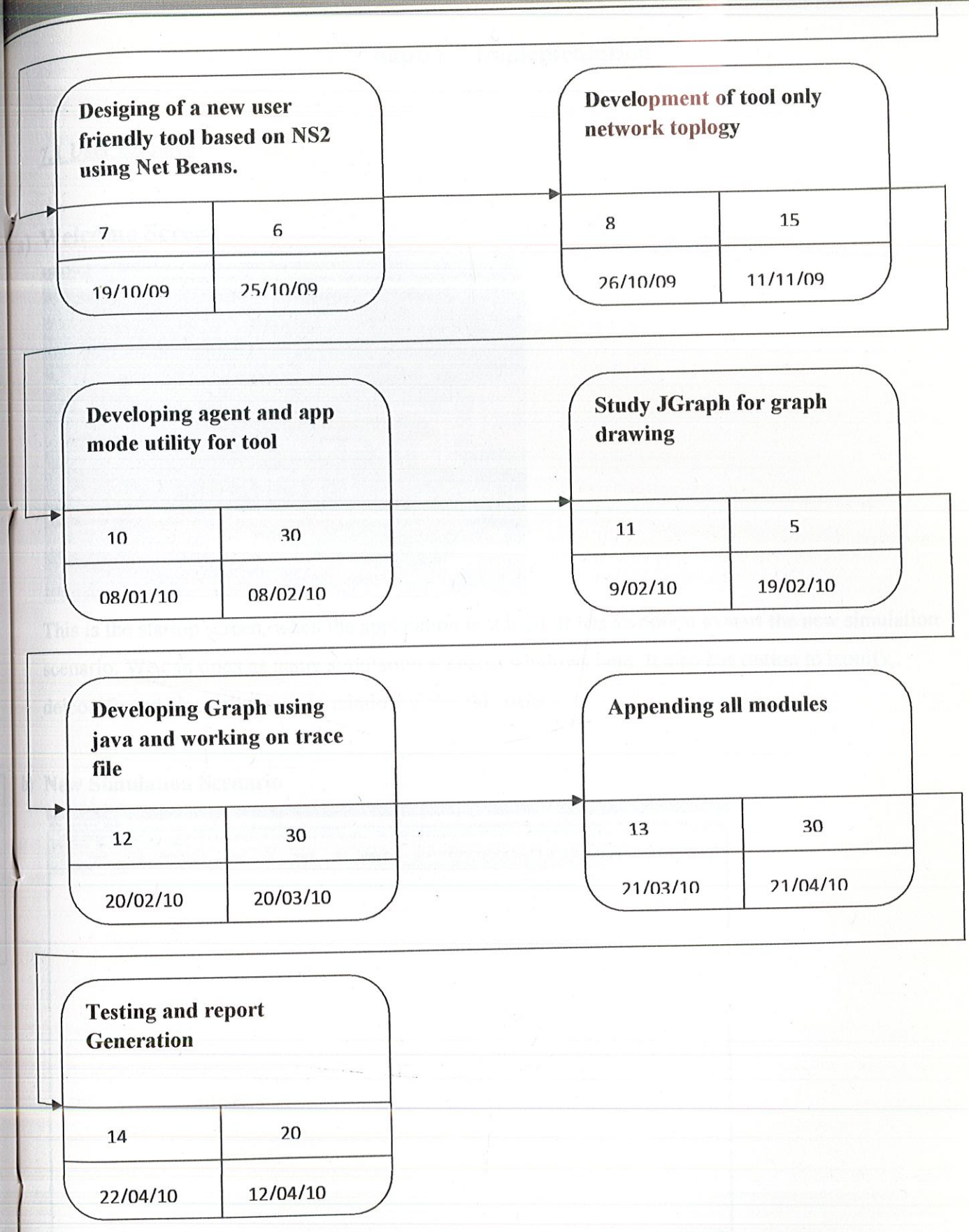
6.3 Modular decomposition of tool



6.4 Project Development Chart

Total days : 275 days




Desiging of a new user friendly tool based on NS2 using Net Beans.
7
6
19/10/09
25/10/09
Development of tool only network toplogy
8
15
26/10/09
11/11/09
Developing agent and app mode utility for tool
10
30
08/01/10
08/02/10
Study JGraph for graph drawing
11
5
9/02/10
19/02/10
Developing Graph using java and working on trace file
12
30
20/02/10
20/03/10
Appending all modules
13
30
21/03/10
21/04/10
Testing and report Generation
14
20
22/04/10
12/04/10

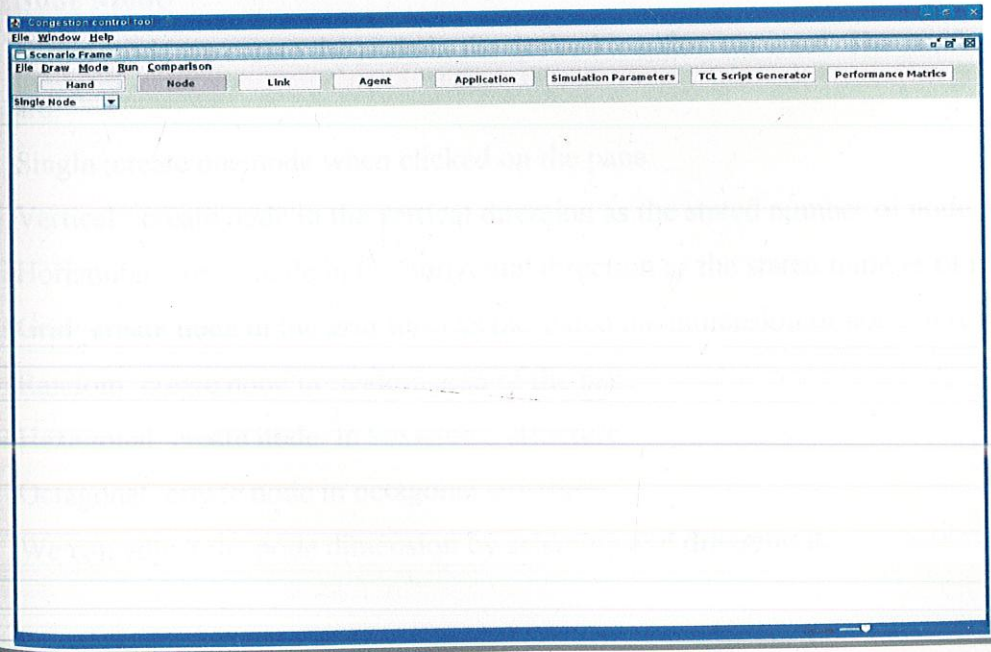
Chapter 7 Implementation

7.1 User Inteface

a) Welcome Screen

This is the startup screen, when the application is started. It has an option to start the new simulation scenario. We can open as many simulation scenario windows here. It also has option to iconify, deiconify and close all the child window under this pane.

b) New Simulation Scenario



This is the main working area where all the features of the tool are available. It has various mode :

1. Hand mode
2. Node mode
3. Link mode
4. Agent mode
5. Application mode

These links are attached in the following way:

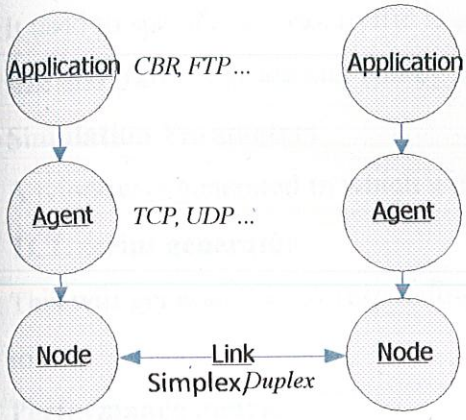


Fig.

Hand Mode

In this we can simply adjust the view of the scenario generated by the user. It is done by dragging the window in the pane the scenario will move according to it.

Node Mode

In this mode we create the node in the desired location we want. The number we can create the nodes are:

1) Single :create one node when clicked on the pane
2) Vertical : create node in the vertical direction as the stated number of node in text field
3) Horizontal: create node in the horizontal direction as the stated number of node in text field
4) Grid: create node in the grid form as the stated the dimension of node in text field
5) Random: create node in random area of the pane
6) Hexagonal: create node in hexagonal structure
7) Octagonal: create node in octagonal structure

We can adjust the node dimension by selecting and dragging it.

Link mode

This mode is used to create the link between the different nodes created in the link mode. We can configure the setting of the link also

Agent Mode

This mode is used to attach agent to the node. The agent may be tcp or udp. We can configure their setting by right click on them. The destination agent get link to the source agent.

Application Mode

It used to specify the way traffic is generate by the agent on a particular node. We can give the packet size and traffic starting and ending time.

Simulation Parameters

A window is generated in which we can set the parameters for the simulation scenario.

TCL script generator

This will generate the corresponding TCL script in new editor pane. This script can be saved and made it to run.

Performance metric

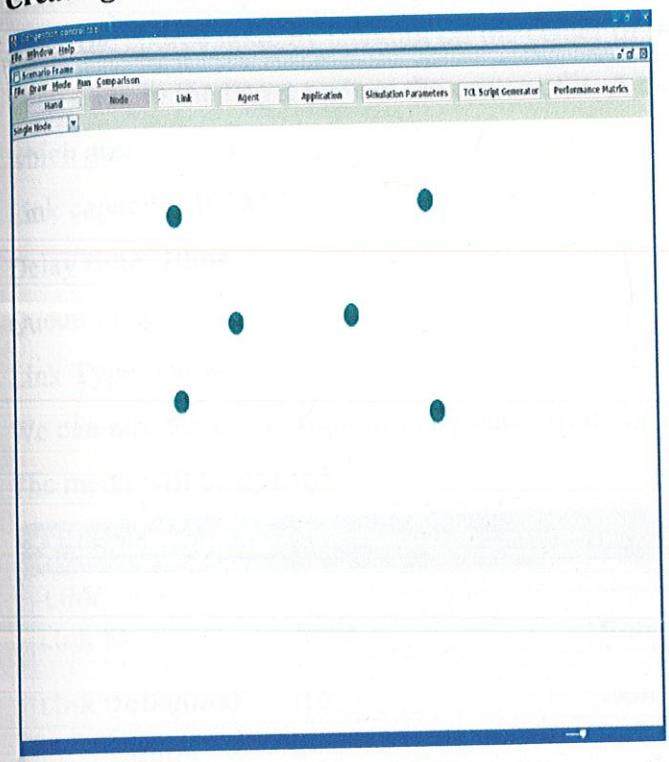
After generating the script, we run it to create the trace file. Using that trace file we can generate the performance metric, which consists of:

1) Delay
2) Packet loss
3) Throughput
4) Queue length
5) Jitter

MENUS

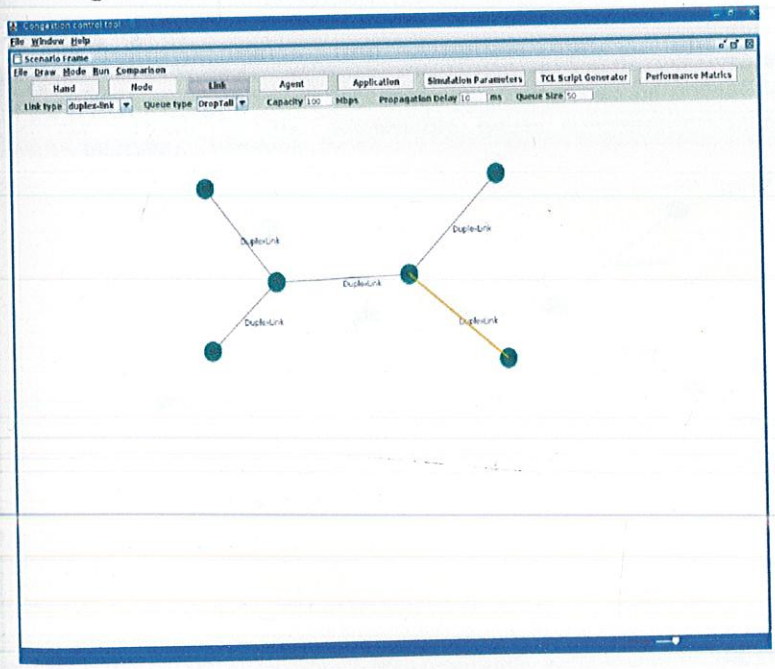
a) **File :** closing the scenario pane
b) **Draw:** checking whether to see specific details in the pane
c) **Mode:** modes can selected using it
d) **Run:** TCL script can be run on the NAM editor
e) **Comparison:** various scenario can compared on the basis of performance metrics

c) **Creating and deleting the node**



The node is created just by clicking on the scenario pane. Generation of can be selected from he dropdown list. For deleting the node we just simply right click on it. A new popup window will be created, which ask for deleting the node. Select it to delete the node.

d) **Creating link**



By clicking on the link mode, we can draw the links between various nodes. To draw it we first select the source node by clicking on the center of the node. There will be orange light around the node. Then click to the target node at the center of it. A new link will be generated with default parameters, which are:

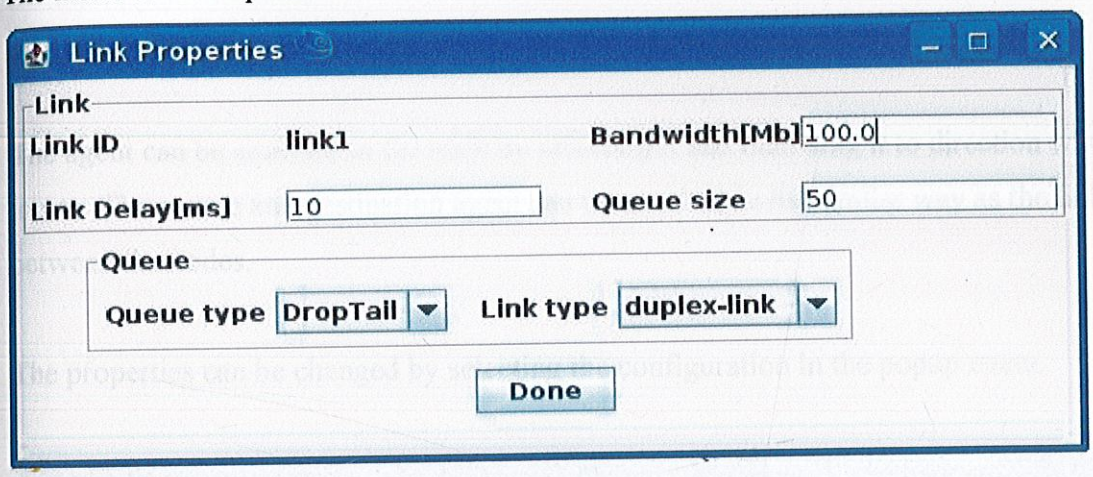
Link capacity: 100MBps

Delay time: 10ms

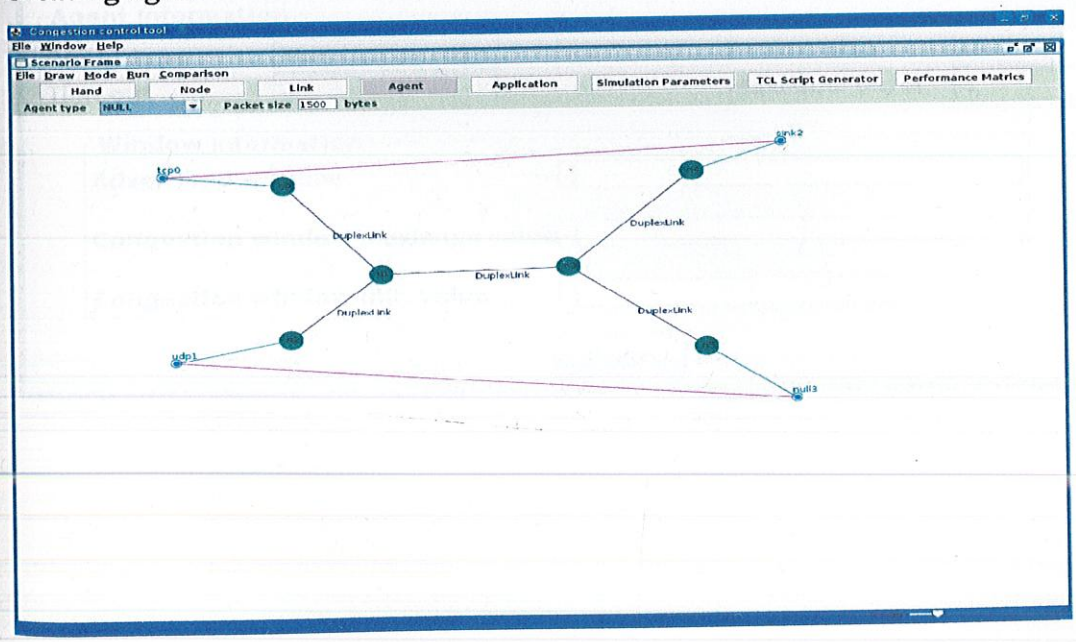
Queue Length: 50

Link Type: Duplex

We can also set its configuration by generating the popup menu, and selecting its configuration setting. The menu will be opened



e) **Creating agent**

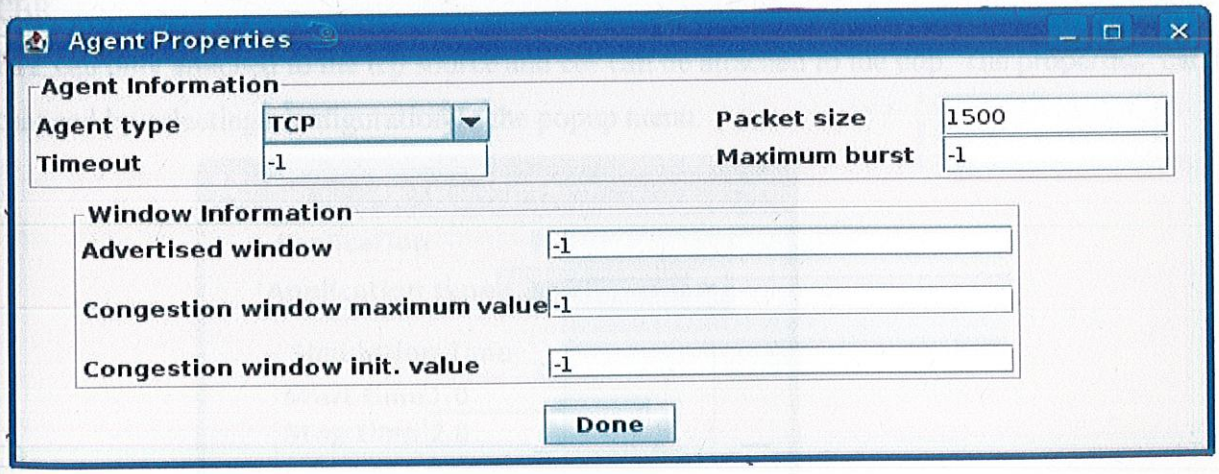


The agent is created by selecting the agent mode. Then select the type of agent you want to create. The type agent can be selected from the dropdown box which has following agent

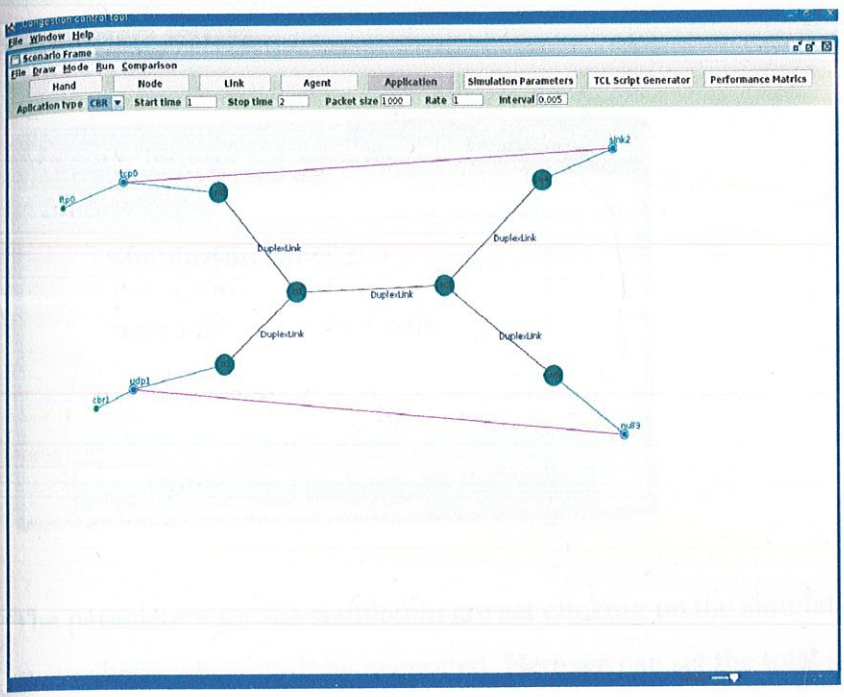
1) TCP
2) TCP/Tahoe
3) TCP/Reno
4) TCP/Newreno
5) TCP/Vegas
6) TCP Sink
7) UDP
8) NULL

The agent can be attached to the node by selecting it and then drag it to direction where you want to locate. The source and destination agent can be attached in the similar way as the link is established between the nodes.

The properties can be changed by selecting the configuration in the popup menu.



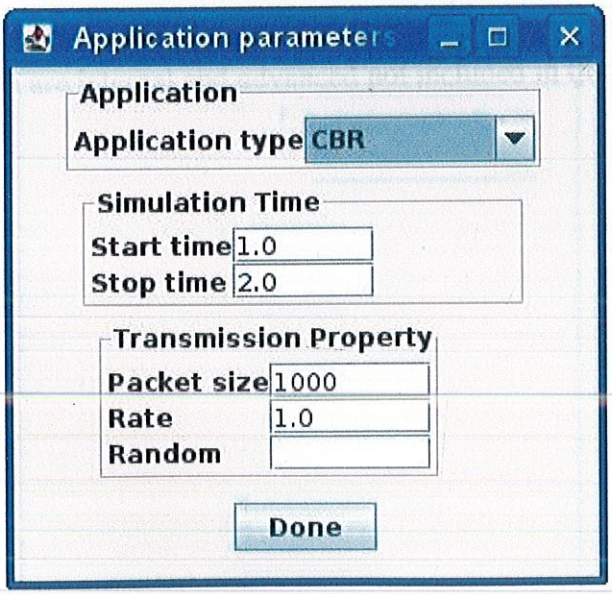
1) Creating application



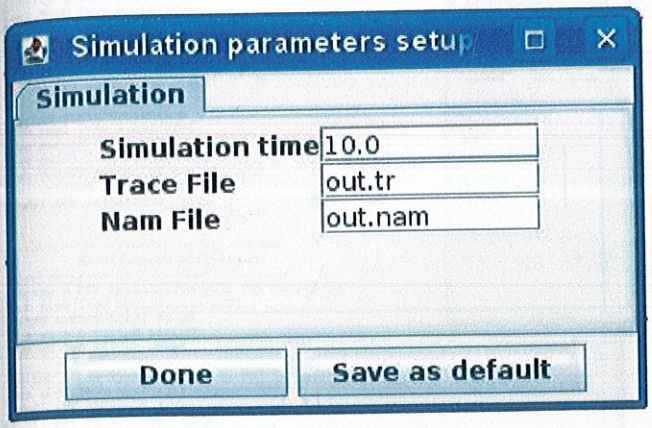
The application is attached to the agent in same fashion as the agent is attached to the node. There are two types of application which can be attached.

1) FTP
2) CBR

FTP can only attached to the tcp source and cbr can be attached to the udp. The properties can be changed by selecting configuration in the popup menu.



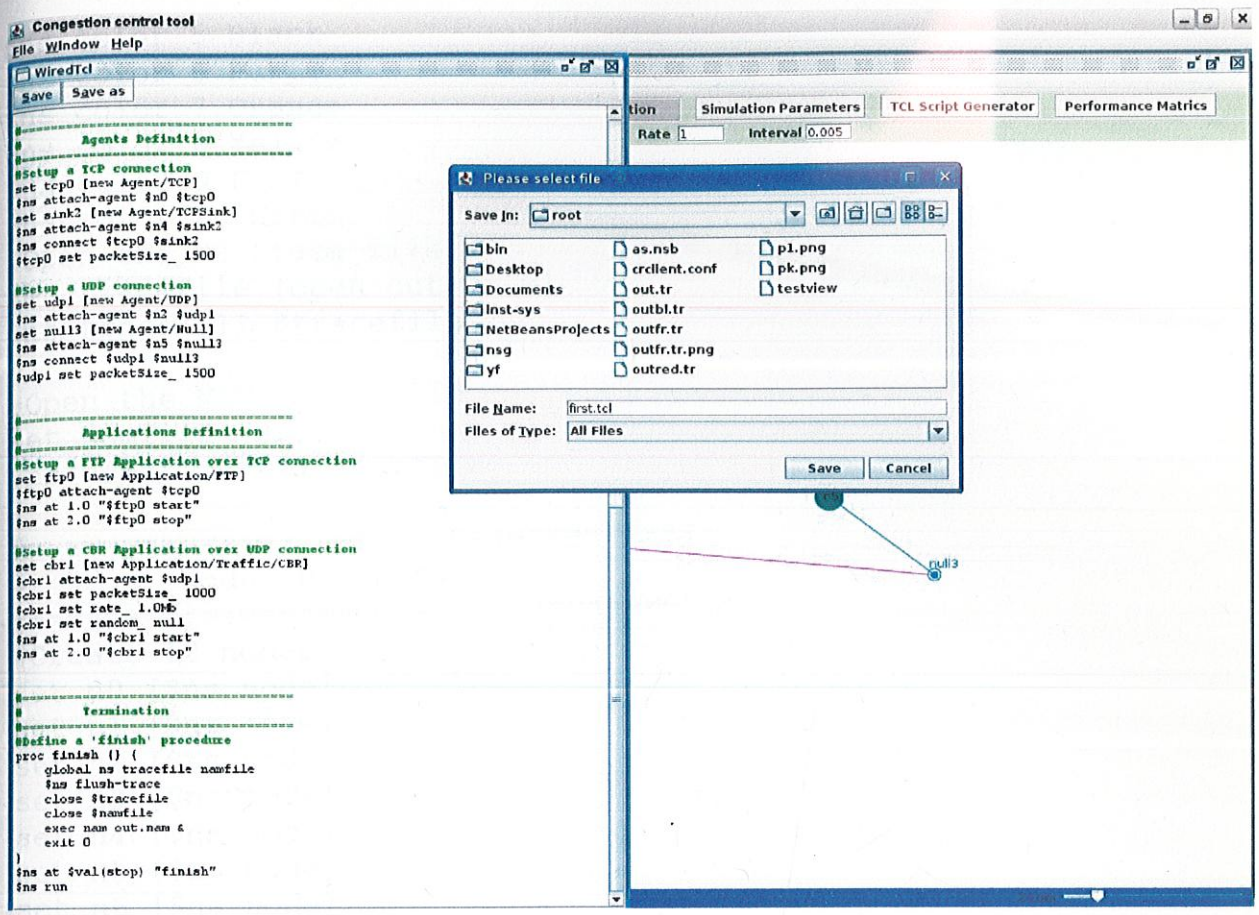
g) **Parameters setting**



The parameters for the simulation are set clicking on the simulation parameters in toolbar. A small menu shown above will be generated. Here we can set the total simulation time, trace file and the NAM file which will be reflected when we will generate the TCL script.

h) **TCL script generation**

This is the most important application of the tool. The TCL script is generated itself by just few clicks away. We can select the button TCL script generator in the toolbar. A new editor pane will be opened with the TCL script printed on it. The green color shows the comment in the TCL script. We can save it into the tcl format by choosing save button in that editor pane. We can also change the other setting in that pane which are internal and advanced not included in the tool.



When you save the script it stored in .tcl file with the filename. The sample script is shown here

```
# This script is generated by automated Tool
# This can be run with help of network simulator2

#===================================
#     Simulation parameters setup
#===================================
set val(stop)   50.0                          ;# total time of
runnning simulation

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

$ns color 1 Blue
$ns color 2 Red
$ns color 3 White
$ns color 4 Brown
```

```
$ns color 5 Black
$ns color 6 Purple
$ns color 7 Orange
$ns color 8 Yellow
$ns color 9 Pink
$ns color 10 Green
#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

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

#=====================================
#        Nodes Definition
#=====================================
#Create 12 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n17 [$ns node]
set n18 [$ns node]
set n19 [$ns node]
set n20 [$ns node]
set n21 [$ns node]

#=====================================
#        Links Definition
#=====================================
#Createlinks between nodes
$ns duplex-link $n0 $n5 100.0Mb 10ms DropTail
$ns queue-limit $n0 $n5 50
$ns duplex-link $n1 $n5 100.0Mb 10ms DropTail
$ns queue-limit $n1 $n5 50
$ns duplex-link $n5 $n3 100.0Mb 10ms DropTail
$ns queue-limit $n5 $n3 50
$ns duplex-link $n4 $n5 100.0Mb 10ms DropTail
$ns queue-limit $n4 $n5 50
$ns duplex-link $n2 $n5 100.0Mb 10ms DropTail
$ns queue-limit $n2 $n5 50
$ns duplex-link $n5 $n6 50.0Mb 40ms RED
$ns queue-limit $n5 $n6 20
```

```
$ns duplex-link $n17 $n6 100.0Mb 10ms DropTail
$ns queue-limit $n17 $n6 50
$ns duplex-link $n17 $n6 100.0Mb 10ms DropTail
$ns queue-limit $n17 $n6 50
$ns duplex-link $n18 $n6 100.0Mb 10ms DropTail
$ns queue-limit $n18 $n6 50
$ns duplex-link $n6 $n19 100.0Mb 10ms DropTail
$ns queue-limit $n6 $n19 50
$ns duplex-link $n6 $n20 100.0Mb 10ms DropTail
$ns queue-limit $n6 $n20 50
$ns duplex-link $n21 $n6 100.0Mb 10ms DropTail
$ns queue-limit $n21 $n6 50

#Give node position (for NAM)
$ns duplex-link-op $n0 $n5 orient right-down
$ns duplex-link-op $n1 $n5 orient right-down
$ns duplex-link-op $n5 $n3 orient left-down
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n2 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right
$ns duplex-link-op $n17 $n6 orient left-down
$ns duplex-link-op $n17 $n6 orient left-down
$ns duplex-link-op $n18 $n6 orient left-down
$ns duplex-link-op $n6 $n19 orient right
$ns duplex-link-op $n6 $n20 orient right-down
$ns duplex-link-op $n21 $n6 orient left-up

#====================================
#        Agents Definition
#====================================
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink5 [new Agent/TCPSink]
$ns attach-agent $n17 $sink5
$ns connect $tcp0 $sink5
$tcp0 set packetSize_ 2000
$tcp0 set window_ 8000
$tcp0 set fid_ 1

#Setup a TCP/FullTcp/Tahoe connection
set tcp1 [new Agent/TCP/FullTcp/Tahoe]
$ns attach-agent $n1 $tcp1
set sink6 [new Agent/TCPSink]
$ns attach-agent $n18 $sink6
$ns connect $tcp1 $sink6
$tcp0 set window_ 8000
```

```
$tcp1 set packetSize_ 2000

$tcp1 set fid_ 2
#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set null7 [new Agent/Null]
$ns attach-agent $n19 $null7
$ns connect $udp2 $null7
$udp2 set packetSize_ 2000

$udp2 set fid_ 3
#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n3 $udp3
set null8 [new Agent/Null]
$ns attach-agent $n20 $null8
$ns connect $udp3 $null8
$udp3 set packetSize_ 1500

$udp3 set fid_ 4
#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n4 $udp4
set null9 [new Agent/Null]
$ns attach-agent $n21 $null9
$ns connect $udp4 $null9
$udp4 set packetSize_ 1500

$udp4 set fid_ 5

#=================================
#        Applications Definition
#=================================
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set type_ FTP
$ns at 2.0 "$ftp0 start"
$ns at 49.0 "$ftp0 stop"

#Setup a FTP Application over TCP/FullTcp/Tahoe connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp0 set type_ FTP
$ns at 8.0 "$ftp1 start"
$ns at 44.0 "$ftp1 stop"
```

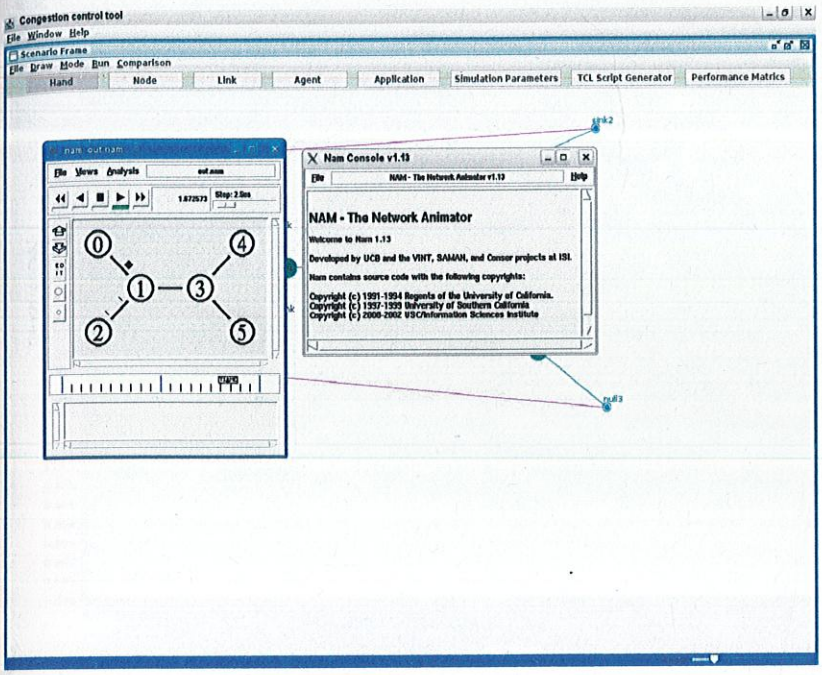
```
#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 1.0Mb
$cbr2 set random_
$ns at 13.0 "$cbr2 start"
$ns at 40.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp3
$cbr3 set packetSize_ 1000
$cbr3 set rate_ 1.0Mb
$cbr3 set random_
$ns at 18.0 "$cbr3 start"
$ns at 35.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 1.0Mb
$cbr4 set random_
$ns at 23.0 "$cbr4 start"
$ns at 33.0 "$cbr4 stop"


#=====================================
#        Termination
#=====================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
$ns at $val(stop) "finish"
$ns run
```

i) **Running the script in NAM editor**

To run a .tcl file with the help of tool, two conditions are necessary:

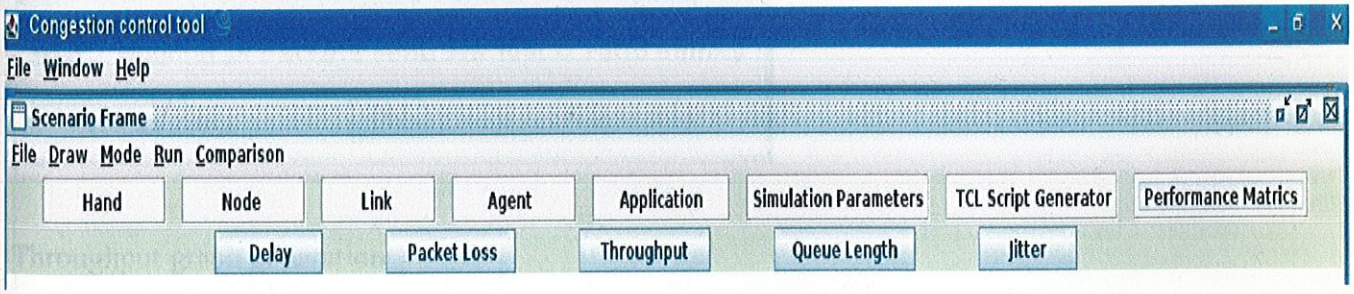
1) NS2 should have been installed previously with the environment variable set

2) NAM should be properly installed previously with the environment variable set



j) **Generating performance metric**

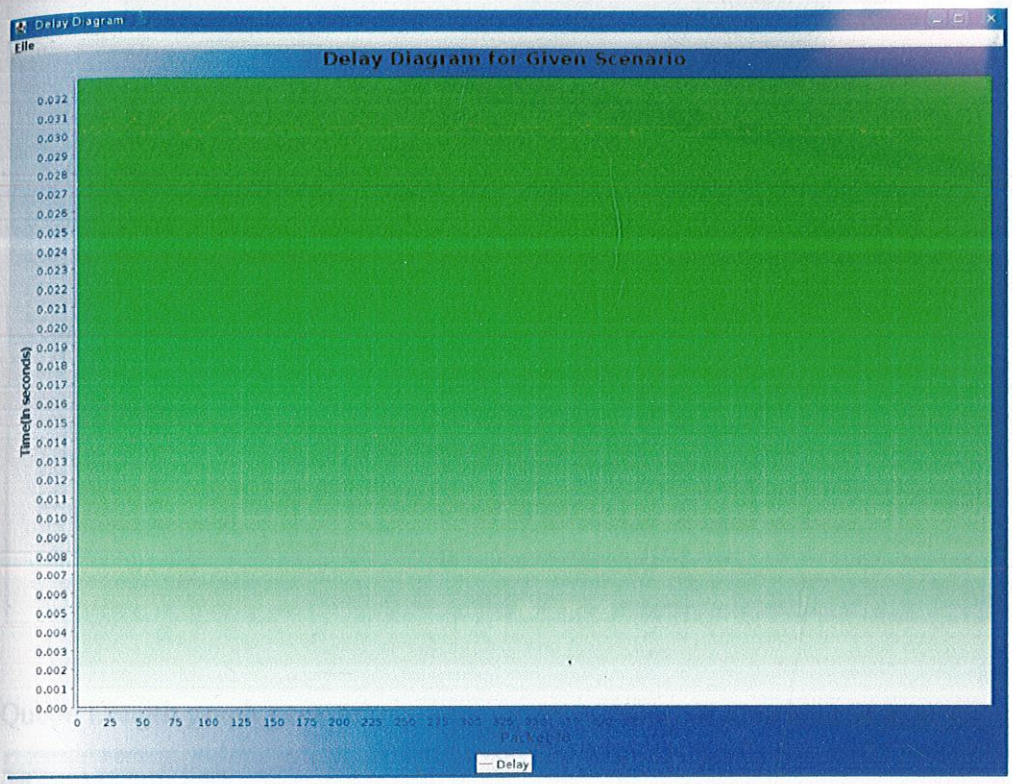
The various performance metrics' can be selected from the button which appears in the panel. They all generated using the algorithm shown in AWK code in chapter 4.



l) calculation delay for tr script

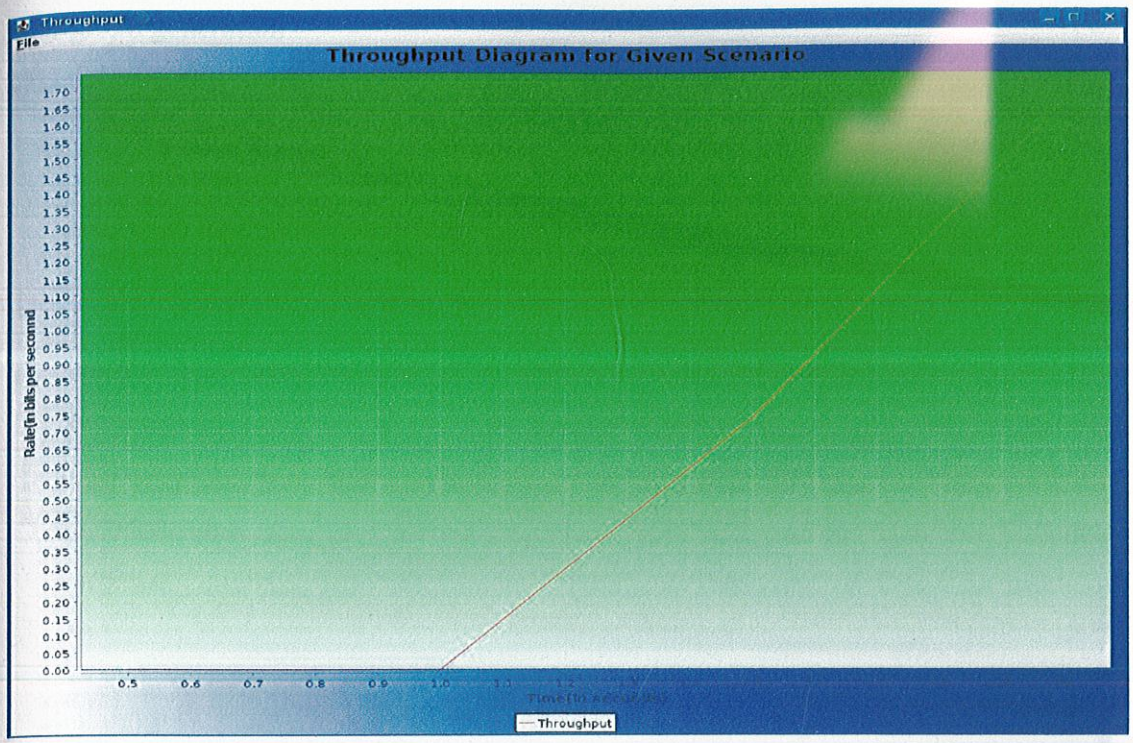
First click on the delay button from the performance metrics panel. New window will be opened to select the trace file for simulated scenario. The graph will be automatically generated. This graph can be

save as "JPEG" format by choose save option from the menu bar of the graph. The sample graph for delay is shown:

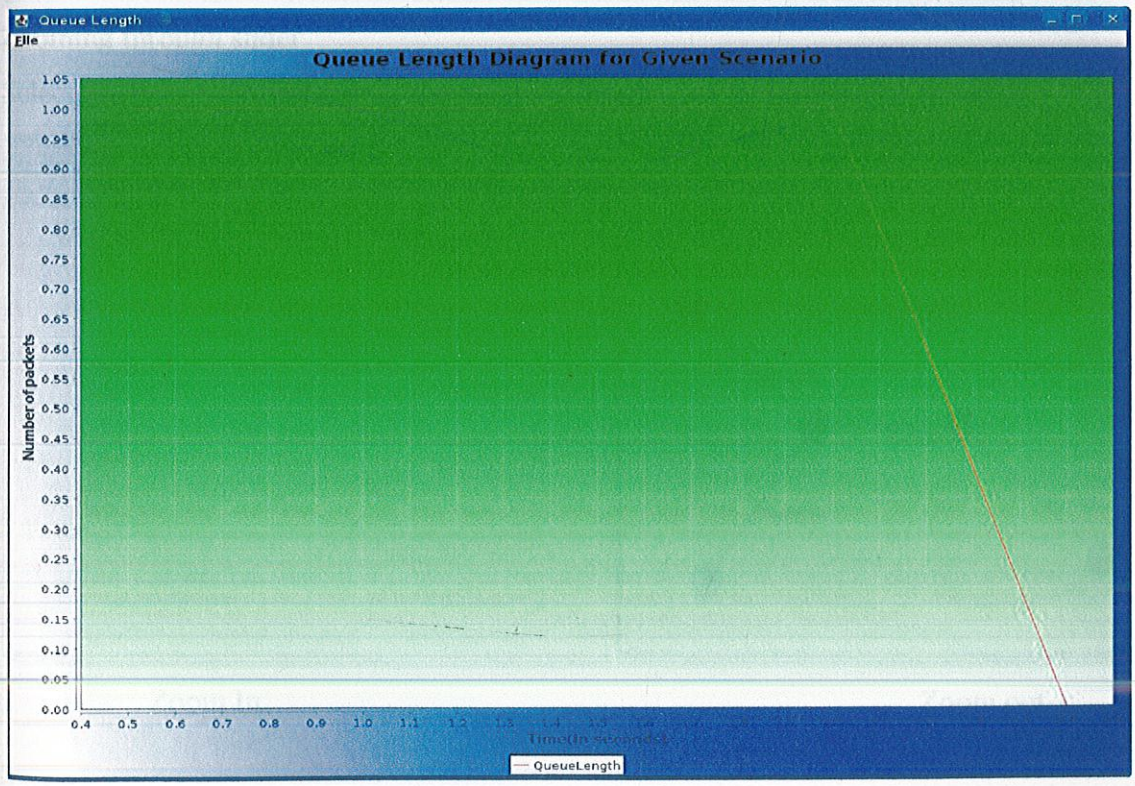


2) Packet Loss calculation

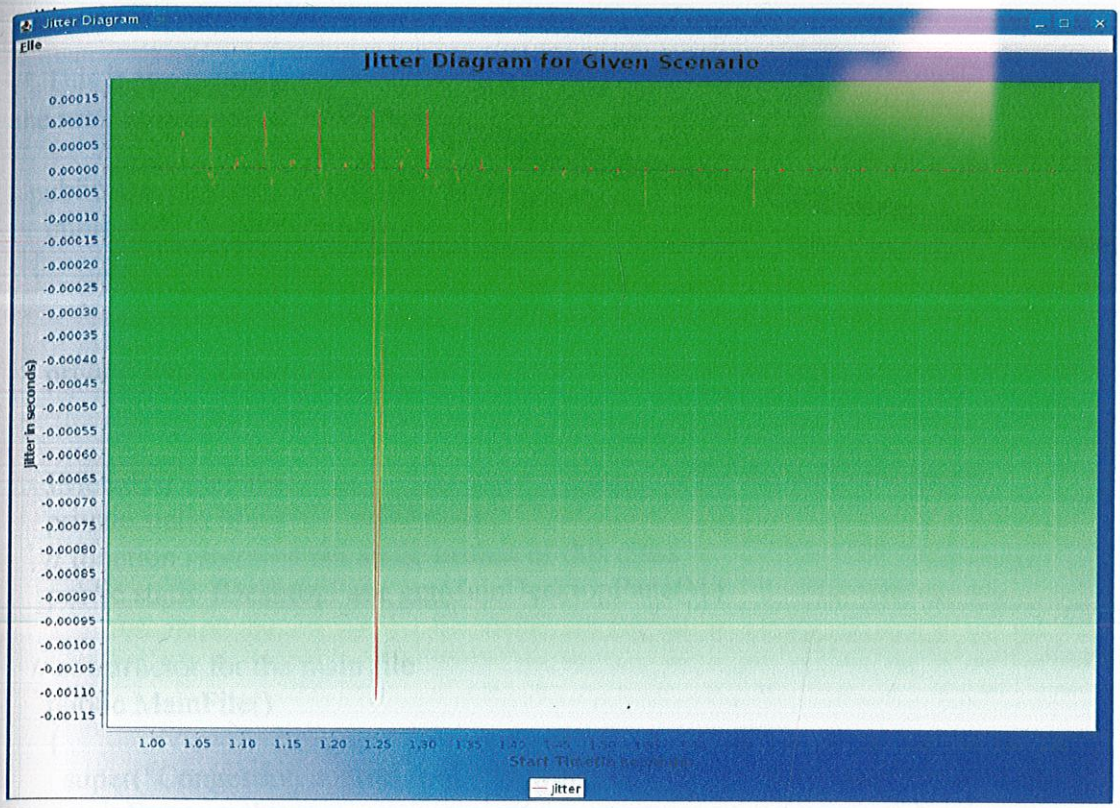
3) Throughput graph generation



4) Queue Length graph generation

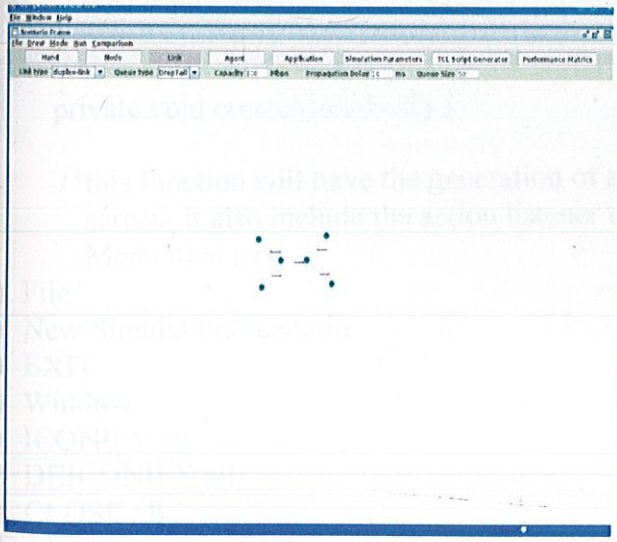


5) Jitter graph generation

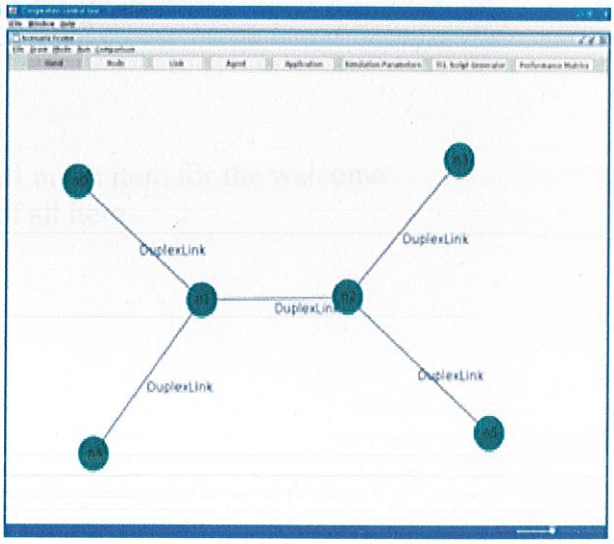


k) Zooming through slider

Select the key in the slider drag the mouse to zoom in or zoom out. For example:



Zoom In



Zoom out

7.2 Source code

MainFile.java

```
/* This is the mainfile of the tool here welcome screen is created. Then it create the scene manager for
the new simulation scenario. */

 public class MainFile extends JFrame
  implements BasicParameters
 {
                public static final int WIRED_FRAME = 1; // constant stating the wired scenario setting

  private static DesktopPaneDraw jdp = new DesktopPaneDraw();//welcome screen


  JMenuBar menubar = new JMenuBar();//menubar for the welcome screen
  private static MainFile mainFrame;
  // function returning the static instant of this class
  public static JDesktopPane getMainDesktopPane(){}

 // constructor for the main file
  public MainFile()
  {
   super("Congestion control tool");//Setting title for welcome screen
   mainFrame = this;
   jdp.setDragMode(1); //setting drag mode on
   getContentPane().add(jdp, "Center");//adding the desktopane to the Jframe
   createMenuBar();//creating menu bar for the main file
  }

   private void createMenuBar() {

    /*this function will have the generation of all menu item for the welcome
      screen. It also include the action listener of all item.
      Menu item has:
1) File
a) New Simulation Scenario
b) EXIT
2) Window
a) ICONIFY all
b) DEICONIFY all
c) CLOSE all
3) About
a) Help*/

    }
   // this is used to create the internal frame for new simulation scenario
```

```
public void createInternalFrame() {
  JInternalFrame scenarioframe = new JInternalFrame();
  scenarioframe = new ScenarioFrame(this);
  scenarioframe.setBounds(0, 0, jdp.getWidth(), jdp.getHeight());
  scenarioframe.setDefaultCloseOperation(2);
  scenarioframe.setVisible(true);

  jdp.add(scenarioframe);

  try
  {
   scenarioframe.setSelected(true);
  } catch (PropertyVetoException e) {
   e.printStackTrace();
  }
 }
 public void createTCLManager(ScenarioFrame sm)
  {
  //call TCL script generation
  }
// function is will run the selectedTCl file
  public void runTcl() {
     //Running the TCL file In nam editor
     int ch;
     File traceFile;
     JFileChooser tracejfc = new JFileChooser();
     try
          {
            tracejfc.setDialogTitle("Please select file");
            int m = tracejfc.showOpenDialog(null);
            if(m == 0)
            {

               traceFile = tracejfc.getSelectedFile();

            } else
            {
               return;
            }
          }
          catch(Exception evt)
          {
            System.out.println(evt.getMessage());
            return;
          }
```

```
    String sctcl ="ns "+traceFile.toString();
            try
                    {
Runtime rt = Runtime.getRuntime();
        Process proc = rt.exec(sctcl);
        int exitVal = proc.exitValue();
System.out.println("Process exitValue: " + exitVal);
            /*Process myProcess = Runtime.getRuntime().exec(sctcl);
              // myProcess=Runtime.getRuntime().exec("echo $?");
                  InputStreamReader myIStreamReader = new
InputStreamReader(myProcess.getInputStream());
                  while ((ch = myIStreamReader.read()) != -1)
                   { System.out.print((char)ch); }*/
                }
           catch (IOException anIOException)
               {
                  System.out.println(anIOException);
                  anIOException.printStackTrace();
                   }
  }


  public static void main(String[] args) {
   MainFile f = new MainFile();

   f.setDefaultCloseOperation(3);
   f.setExtendedState(6);
   f.setVisible(true);
  }


}
```

ScenarioFrame.java

```
// to generate the new scenario Frame
 public class ScenarioFrame extends JInternalFrame
  implements BasicParameters
 {
  static final long serialVersionUID = 0L;
  public boolean drawNode = true;
  public boolean drawLink = true;
  public boolean drawLinkDetail = false;
  public boolean drawAgent = true;
  public boolean drawAgentDetail = false;
  public boolean drawApp = true;
```

```
public JButton b1 = new JButton("       Hand        ");
public JButton b2 = new JButton("       Node        ");
public JButton b3 = new JButton("       Link        ");
public JButton b4 = new JButton("      Agent        ");
public JButton b5 = new JButton("    Application    ");
public JButton b6 = new JButton("Simulation Parameters");
public JButton b7 = new JButton(" TCL Script Generator");
public JButton b8 = new JButton(" Performance Matrics ");

JMenuBar menubar = new JMenuBar();
public float centerX = -1.0F;
public float centerY = -1.0F;
JToolBar toolbar = new JToolBar();
MainFile nsg;
JPanel center = new JPanel(new BorderLayout());
public JLabel status = new JLabel();
JSlider slider = new JSlider(8, 500, 100);
SimulationParamterDialog parameters;
public DataStorage dm;
public VisualizeScene sv;
public LinkMenu linkmenu = new LinkMenu(this.nsg);
public AgentMenu agentmenu = new AgentMenu(this.nsg);
public AppMenu appmenu = new AppMenu(this.nsg);
public DelayMenuComparison dmcmp;
public PacketLossComparison pcktcmp;
public ThroughputComparison thrcmp;
public QueueLengthComparison qlcmp;
public JitterComparison jtrcmp;

public int goEast = 0;
public int goWest = 0;
public int goNorth = 0;
public int goSouth = 0;


public Thread thread;

public boolean scroll(int x, int y)
{
 return false;
}

public ScenarioFrame(MainFile f)
{
 super("", true, true, true, true);
 this.nsg = f;
```

```
    this.dm = new DataStorage();
    this.sv = new VisualizeScene(this);

    setTitle("Scenario Frame");
    this.parameters = new SimulationParamterDialog(this.nsg);
    createMenuBar();
    createToolBar();

    Box p = Box.createHorizontalBox();
    this.status.setPreferredSize(new Dimension(200, 20));
    this.status.setMaximumSize(new Dimension(200, 20));
    this.status.setMinimumSize(new Dimension(200, 20));

    this.slider.setPreferredSize(new Dimension(200, 20));
    this.slider.setMaximumSize(new Dimension(200, 20));
    this.slider.setMinimumSize(new Dimension(200, 20));
    this.slider.addChangeListener(new ChangeListener() {
      public void stateChanged(ChangeEvent e) {
        if (ScenarioFrame.this.centerX == -1.0F) {
          ScenarioFrame.this.centerX = (ScenarioFrame.this.sv.getWidth() / 2 /
ScenarioFrame.this.sv.scale - ScenarioFrame.this.sv.shiftX);
          ScenarioFrame.this.centerY = (ScenarioFrame.this.sv.getHeight() / 2 /
ScenarioFrame.this.sv.scale - ScenarioFrame.this.sv.shiftY);
        }
        ScenarioFrame.this.sv.scale = (ScenarioFrame.this.slider.getValue() / 100.0F);
        ScenarioFrame.this.sv.shiftX = (int)(ScenarioFrame.this.sv.getWidth() / 2 /
ScenarioFrame.this.sv.scale - ScenarioFrame.this.centerX);
        ScenarioFrame.this.sv.shiftY = (int)(ScenarioFrame.this.sv.getHeight() / 2 /
ScenarioFrame.this.sv.scale - ScenarioFrame.this.centerY);
        ScenarioFrame.this.sv.repaint();
      }
    });


    p.setBackground(STATUS_LABEL_COLOR);
    p.setOpaque(true);
    this.slider.setOpaque(false);
    p.add(Box.createHorizontalGlue());
    p.add(new JLabel("Zoom"));
    p.add(this.slider);
    this.center.add(p, "South");
    this.center.add(this.sv, "Center");
    getContentPane().add(this.center, "Center");

  }
```

```
public void createToolBar() {
 this.toolbar.setBackground(TOOLBAR_COLOR);
 this.toolbar.setOpaque(true);
 this.toolbar.setFloatable(false);
 getContentPane().add(this.toolbar, "North");
 this.toolbar.add(Box.createHorizontalStrut(35));
 this.b1.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
   ScenarioFrame.this.sv.switchMode(1);
  }
 });
 this.toolbar.add(this.b1);
 this.toolbar.add(Box.createHorizontalStrut(15));
 this.b2.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
   ScenarioFrame.this.sv.switchMode(2);
  }
 });
 this.toolbar.add(this.b2);
 this.toolbar.add(Box.createHorizontalStrut(15));

 this.b3.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    ScenarioFrame.this.sv.switchMode(5);
  }
 });
 this.toolbar.add(this.b3);
 this.toolbar.add(Box.createHorizontalStrut(15));

 this.b4.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
   ScenarioFrame.this.sv.switchMode(3);
  }
 });
 this.toolbar.add(this.b4);
 this.toolbar.add(Box.createHorizontalStrut(15));
 this.b5.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
   ScenarioFrame.this.sv.switchMode(4);

  }
 });
 this.toolbar.add(this.b5);
 this.toolbar.add(Box.createHorizontalStrut(15));
 this.b6.addActionListener(new ActionListener(){
  public void actionPerformed(ActionEvent e) {
```

```
        ScenarioFrame.this.parameters.setVisible(true);
      }
    });
    this.toolbar.add(this.b6);
    this.toolbar.add(Box.createHorizontalStrut(15));

    this.b7.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e) {
        ScenarioFrame.this.nsg.createTCLManager(ScenarioFrame.this);
      }
    });
    this.toolbar.add(this.b7);
    this.toolbar.add(Box.createHorizontalStrut(15));
    this.b8.addActionListener(new ActionListener(){
      public void actionPerformed(ActionEvent e) {
        ScenarioFrame.this.sv.switchMode(6);
      }
    });
    this.toolbar.add(this.b8);
  }

  public void createMenuBar() {
    setJMenuBar(this.menubar);

    JMenu menu = new JMenu("File");
    menu.setMnemonic('F');

    JMenuItem item = new JMenuItem("Close");
    item.setMnemonic('C');
    item.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        ScenarioFrame.this.dispose();
      }
    });
    menu.add(item);

    this.menubar.add(menu);

    menu = new JMenu("Draw");
    menu.setMnemonic('D');

    JCheckBoxMenuItem checkItem = new JCheckBoxMenuItem("Draw node");
    checkItem.setAccelerator(KeyStroke.getKeyStroke(113, 0));
    checkItem.setSelected(true);
    checkItem.addItemListener(new ItemListener() {
      public void itemStateChanged(ItemEvent evt) {
```

```
    if (evt.getStateChange() == 1)
      ScenarioFrame.this.drawNode = true;
    else {
      ScenarioFrame.this.drawNode = false;
    }
    ScenarioFrame.this.repaint();
  }
});
menu.add(checkItem);
checkItem = new JCheckBoxMenuItem("Draw link");
checkItem.setAccelerator(KeyStroke.getKeyStroke(114, 0));
checkItem.setSelected(true);
checkItem.addItemListener(new ItemListener() {
   public void itemStateChanged(ItemEvent evt) {
      if (evt.getStateChange() == 1)
         ScenarioFrame.this.drawLink = true;
      else {
         ScenarioFrame.this.drawLink = false;
      }
      ScenarioFrame.this.repaint();
   }
});
menu.add(checkItem);
checkItem = new JCheckBoxMenuItem("Draw link detail");
checkItem.setAccelerator(KeyStroke.getKeyStroke(115, 0));
checkItem.setSelected(false);
checkItem.addItemListener(new ItemListener() {
   public void itemStateChanged(ItemEvent evt) {
      if (evt.getStateChange() == 1)
         ScenarioFrame.this.drawLinkDetail = true;
      else {
         ScenarioFrame.this.drawLinkDetail = false;
      }
      ScenarioFrame.this.repaint();
   }
});
menu.add(checkItem);
checkItem = new JCheckBoxMenuItem("Draw agent");
checkItem.setAccelerator(KeyStroke.getKeyStroke(116, 0));
checkItem.setSelected(true);
checkItem.addItemListener(new ItemListener() {
  public void itemStateChanged(ItemEvent evt) {
    if (evt.getStateChange() == 1)
      ScenarioFrame.this.drawAgent = true;
    else {
      ScenarioFrame.this.drawAgent = false;
```

```
    }
    ScenarioFrame.this.repaint();
  }
});
menu.add(checkItem);

checkItem = new JCheckBoxMenuItem("Draw agent detail");
checkItem.setAccelerator(KeyStroke.getKeyStroke(117, 0));
checkItem.setSelected(false);
checkItem.addItemListener(new ItemListener() {
  public void itemStateChanged(ItemEvent evt) {
    if (evt.getStateChange() == 1)
      ScenarioFrame.this.drawAgentDetail = true;
    else {
      ScenarioFrame.this.drawAgentDetail = false;
    }
    ScenarioFrame.this.repaint();
  }
});
menu.add(checkItem);

checkItem = new JCheckBoxMenuItem("Draw application");
checkItem.setAccelerator(KeyStroke.getKeyStroke(118, 0));
checkItem.setSelected(true);
checkItem.addItemListener(new ItemListener() {
  public void itemStateChanged(ItemEvent evt) {
    if (evt.getStateChange() == 1)
      ScenarioFrame.this.drawApp = true;
    else {
      ScenarioFrame.this.drawApp = false;
    }
    ScenarioFrame.this.repaint();
  }
});
menu.add(checkItem);

this.menubar.add(menu);
 menu = new JMenu("Mode");
menu.setMnemonic('M');
item = new JMenuItem("Hand mode");
item.setAccelerator(KeyStroke.getKeyStroke(49, 2));
item.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    ScenarioFrame.this.sv.switchMode(1);
  }
});
```

```
menu.add(item);

item = new JMenuItem("Node mode");
item.setAccelerator(KeyStroke.getKeyStroke(50, 2));
item.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    ScenarioFrame.this.sv.switchMode(2);
  }
});
menu.add(item);


item = new JMenuItem("Link mode");
item.setAccelerator(KeyStroke.getKeyStroke(51, 2));
item.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
     ScenarioFrame.this.linkmenu.setVisible(true);
   }
});
menu.add(item);


item = new JMenuItem("Agent mode");
item.setAccelerator(KeyStroke.getKeyStroke(52, 2));
item.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    ScenarioFrame.this.agentmenu.setVisible(true);
  }
});
menu.add(item);

item = new JMenuItem("Application mode");
item.setAccelerator(KeyStroke.getKeyStroke(53, 2));
item.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    ScenarioFrame.this.appmenu.setVisible(true);
  }
});
menu.add(item);

item = new JMenuItem("Parameter");
item.setAccelerator(KeyStroke.getKeyStroke(54, 2));
item.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    ScenarioFrame.this.parameters.setVisible(true);
  }
```

```
});
menu.add(item);

item = new JMenuItem("Generate TCL");
item.setAccelerator(KeyStroke.getKeyStroke(55, 2));
item.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    ScenarioFrame.this.nsg.createTCLManager(ScenarioFrame.this);
  }
});
menu.add(item);

this.menubar.add(menu);
menu = new JMenu("Run");
menu.setMnemonic('R');
item = new JMenuItem("Select script");

item.setAccelerator(KeyStroke.getKeyStroke(57, 2));
item.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    ScenarioFrame.this.nsg.runTcl();
  }
});
menu.add(item);
this.menubar.add(menu);
menu = new JMenu("Comparison");
menu.setMnemonic('C');
item = new JMenuItem("Delay");

 item.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    ScenarioFrame.this.dmcmp=new DelayMenuComparison();
  }
});
menu.add(item);
this.menubar.add(menu);
 item = new JMenuItem("PacketLoss");


item.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent evt) {
    ScenarioFrame.this.pcktcmp=new PacketLossComparison();
  }
});
menu.add(item);
this.menubar.add(menu);
```

```
    item = new JMenuItem("Throughput");

    item.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent evt) {
        ScenarioFrame.this.thrcmp=new ThroughputComparison();
      }
    });
    menu.add(item);
    this.menubar.add(menu);
     item = new JMenuItem("QueueLength");

    item.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent evt) {
        ScenarioFrame.this.qlcmp=new QueueLengthComparison();
      }
    });
    menu.add(item);
    this.menubar.add(menu);
     item = new JMenuItem("Jitter");

    item.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent evt) {
        ScenarioFrame.this.jtrcmp=new JitterComparison();
      }
    });
    menu.add(item);
    this.menubar.add(menu);

  }
}
```

VisualizeScene.java

```
// This class will handle all the interface function of the tool in scenario Frame . it redraws the
//component using the pain() function inherited from parent class
 package nsg;

 import java.awt.BasicStroke;
 import java.awt.Graphics;
 import java.awt.Graphics2D;
 import java.awt.RenderingHints;
 import java.awt.event.MouseWheelEvent;
 import java.awt.event.MouseWheelListener;
 import java.awt.geom.*;
 import javax.swing.JComponent;
```

```
import javax.swing.JPanel;
import nsg.component.Agent;
import nsg.component.App;
import nsg.component.Link;
import nsg.component.Node;
import nsg.interactive.AgentModeHandler;
import nsg.interactive.ApplicationModeHandler;
import nsg.interactive.LinkModeHandler;
import nsg.interactive.CreatingNodeModeHandler;
import nsg.interactive.HandModeHandler;
import nsg.interactive.NormalModeHandler;
import nsg.panels.AgentPanel;
import nsg.panels.AppPanel;
import nsg.panels.HandModePanel;
import nsg.panels.LinkPanel;
import nsg.panels.NodePanel;
import nsg.panels.NormalModePanel;
import nsg.panels.PerformanceMatric;

public class VisualizeScene extends JComponent
 implements BasicParameters
{
 static final long serialVersionUID = 0L;
 public int mainMode;
 public int shiftX;
 public int shiftY;

 public HandModeHandler handModeHandler;
 public CreatingNodeModeHandler creatingNodeModeHandler;
 public NormalModeHandler normalModeHandler;
 public LinkModeHandler creatingLinkModeHandler;
 public AgentModeHandler creatingAgentModeHandler;
 public ApplicationModeHandler creatingAppModeHandler;
 public JPanel modePanel;
 public LinkPanel linkPanel = new LinkPanel(this);
 public NodePanel nodePanel = new NodePanel(this);
 public AgentPanel agentPanel = new AgentPanel(this);
 public AppPanel appPanel = new AppPanel(this);
 public NormalModePanel normalModePanel = new NormalModePanel();
 public HandModePanel handModePanel = new HandModePanel();
 public PerformanceMatric matricModelPanel = new PerformanceMatric(this);
 public float scale = 1.0F;
 public ScenarioFrame sm;
 DataStorage dm;
 public VisualizeScene(ScenarioFrame sm)
 {
```

```
    this.sm = sm;
    sm.sv = this;
    this.dm = sm.dm;
    this.handModeHandler = new HandModeHandler(sm);
    this.creatingNodeModeHandler = new CreatingNodeModeHandler(sm);
    this.creatingLinkModeHandler = new LinkModeHandler(sm);
    this.creatingAgentModeHandler = new AgentModeHandler(sm);
    this.creatingAppModeHandler = new ApplicationModeHandler(sm);
    this.shiftX = -5000;
    this.shiftY = -5000;
    switchMode(2);
    addMouseWheelListener(new MouseWheelListener() {
     public void mouseWheelMoved(MouseWheelEvent e) {
      VisualizeScene.this.sm.slider.setValue(VisualizeScene.this.sm.slider.getValue() +
e.getWheelRotation() * 10);
     }
    });
   }

   public void switchMode(int mode) {
    this.mainMode = mode;
    if (getMouseListeners().length != 0) {
     removeMouseListener(getMouseListeners()[0]);
     removeMouseMotionListener(getMouseMotionListeners()[0]);
    }
    if (this.modePanel != null) {
     this.sm.center.remove(this.modePanel);
     this.sm.validate();
    }
    this.sm.b1.setBackground(BUTTON_DISABLE_COLOR);
    this.sm.b2.setBackground(BUTTON_DISABLE_COLOR);
    this.sm.b3.setBackground(BUTTON_DISABLE_COLOR);
    this.sm.b4.setBackground(BUTTON_DISABLE_COLOR);
    this.sm.b5.setBackground(BUTTON_DISABLE_COLOR);

    switch (mode)
    {
    case 0:
     break;
    case 1:
     this.sm.b1.setBackground(BUTTON_ENABLE_COLOR);
     this.sm.b1.setBounds(0, 0, 200, 50);
     this.modePanel = this.handModePanel;
     addMouseMotionListener(this.handModeHandler);
     addMouseListener(this.handModeHandler);
     break;
```

```
case 2:
 this.sm.b2.setBackground(BUTTON_ENABLE_COLOR);
 this.modePanel = this.nodePanel;
 addMouseMotionListener(this.creatingNodeModeHandler);
 addMouseListener(this.creatingNodeModeHandler);
 break;
case 3:
 this.sm.b4.setBackground(BUTTON_ENABLE_COLOR);

 this.modePanel = this.agentPanel;

 addMouseMotionListener(this.creatingAgentModeHandler);
 addMouseListener(this.creatingAgentModeHandler);
 break;
case 4:
 this.sm.b5.setBackground(BUTTON_ENABLE_COLOR);

 this.modePanel = this.appPanel;
 addMouseMotionListener(this.creatingAppModeHandler);
 addMouseListener(this.creatingAppModeHandler);
 break;
case 5:
 this.sm.b3.setBackground(BUTTON_ENABLE_COLOR);
 System.out.println(this.sm.b5.getBackground());
 this.modePanel = this.linkPanel;
 addMouseMotionListener(this.creatingLinkModeHandler);
 addMouseListener(this.creatingLinkModeHandler);
 break;
case 6:
      this.modePanel=this.matricModelPanel;
 break;
default:
 this.modePanel = this.normalModePanel;

 System.err.println("Mode switching error!!!");
 }
 this.sm.center.add(this.modePanel, "North");
 this.sm.validate();
}

public void paintComponent(Graphics g2) {
 Graphics2D g = (Graphics2D)g2;

 g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
 g.setColor(BACKGROUND_COLOR);
```

```
g.scale(this.scale, this.scale);
g.fillRect(0, 0, 10000, 10000);

g.translate(this.shiftX, this.shiftY);


if (this.sm.drawLink) drawLinks(g);
if (this.sm.drawApp) drawApp(g);
if (this.sm.drawAgent) drawAgents(g);
if (this.sm.drawNode) drawNodes(g);
switch (this.mainMode)
{
case 0:
  break;
case 1:
  break;
case 2:
  this.creatingNodeModeHandler.draw(g);
  break;
case 3:
  this.creatingAgentModeHandler.draw(g);
  break;
case 4:
  this.creatingAppModeHandler.draw(g);
  break;
case 5:
this.creatingLinkModeHandler.draw(g);
  }
}

private void drawInfo(Graphics2D g)
{
 g.translate(-this.shiftX, -this.shiftY);
 g.setColor(INFO_COLOR);
 g.drawString("Number of nodes: " + this.dm.getNodes().length, 20, 20); }

private void drawNodes(Graphics2D g)
{
 int R = 30;

 int Rdiv2 = R / 2;

 Object[] nodes = this.dm.getNodes();
 g.setStroke(new BasicStroke(1.0F, 1, 2));
 g.setColor(NODE_COLOR);
 for (int i = 0; i < nodes.length; ++i) {
```

```
    Node p = (Node)nodes[i];
    Ellipse2D shap = new Ellipse2D.Double(p.x - Rdiv2, p.y - Rdiv2, R, R);

    g.setColor(NODE_COLOR);
    g.fill(shap);
    g.setColor(NODE_TEXT_COLOR);
    if (p.id < 10)
      g.drawString("n" + String.valueOf(p.id), p.x - 6, p.y + 4);
    else if (p.id < 100)
      g.drawString("n" + String.valueOf(p.id), p.x - 10, p.y + 4);
    else {
      g.drawString("n" + String.valueOf(p.id), p.x - 14, p.y + 4);
    }

  }
}
  private void drawLinks(Graphics2D g)
  {
    g.setStroke(new BasicStroke(1.0F, 1, 2));
    g.setColor(LINK_COLOR);

    String type = "";

    Object[] links = this.dm.getLinks();
    for (int i = 0; i < links.length; ++i) {
      Link link = (Link)links[i];
      int X1 = link.src.x;
      int Y1 = link.src.y;
      int X2 = link.dst.x;
      int Y2 = link.dst.y;
      Line2D.Double line = new Line2D.Double(X1, Y1, X2, Y2);
      g.draw(line);
      if (link.linkType == 0)
        type = "DuplexLink";
      else {
        type = "SimplexLink [" + link.src.id + "->" + link.dst.id + "]";
      }
      g.drawString(type, (X1 + X2) / 2, (Y1 + Y2) / 2 + 13);
      if (this.sm.drawLinkDetail)
        if (link.queueSize != -1)
          g.drawString("capacity:" + link.capacity + " propagationDelay:" + link.propagationDelay + "
queueSize:" + link.queueSize + " queueType:" + link.queueType, (X1 + X2) / 2, (Y1 + Y2) / 2 + -11);
        else
          g.drawString("capacity:" + link.capacity + " propagationDelay:" + link.propagationDelay + "
queueType:" + link.queueType, (X1 + X2) / 2, (Y1 + Y2) / 2 + -11);
    }
```

```
}

private void drawAgents(Graphics2D g)
{
 int R = 8;

 int Rdiv2 = R / 2;

 g.setStroke(new BasicStroke(1.0F, 1, 2));

 Object[] agents = this.dm.getAgents();
 for (int i = 0; i < agents.length; ++i) {
  Agent a = (Agent)agents[i];
  g.setColor(AGENT_COLOR);
  Ellipse2D shap = new Ellipse2D.Double(a.x - Rdiv2, a.y - Rdiv2, R, R);
  g.fill(shap);
  shap = new Ellipse2D.Double(a.x - Rdiv2 - 3, a.y - Rdiv2 - 3, R + 5, R + 5);
  g.draw(shap);
  switch (a.agentType)
  {
  case 0:
  case 4:
  case 8:
  case 12:
  case 16:
   if (a.remoteAgent != null) {
    int X1 = a.x;
    int Y1 = a.y;
    int X2 = a.remoteAgent.x;
    int Y2 = a.remoteAgent.y;
    g.setColor(AGENT_LINK_COLOR);
    Line2D.Double line = new Line2D.Double(X1, Y1, X2, Y2);
    g.draw(line);
   }
   g.setColor(AGENT_COLOR);
   g.drawString("tcp" + String.valueOf(a.id), a.x - R + 2, a.y - R);
   if (this.sm.drawAgentDetail) {
    if (a.packetSize != -1)
     g.drawString(Agent.convertType(a.agentType) + " size:" + a.packetSize, a.x - R + 2, a.y - (3 *
R));
    else {
     g.drawString(Agent.convertType(a.agentType) + " size:" + a.packetSize, a.x - R + 2, a.y - (3 *
R));
    }
   }
   break;
```

```
        case 1:
          g.setColor(AGENT_COLOR);
          g.drawString("sink" + String.valueOf(a.id), a.x - R + 2, a.y - R);
          if (this.sm.drawAgentDetail) g.drawString(Agent.convertType(a.agentType), a.x - R + 2, a.y - (3 *
R));
          break;
        case 2:
          if (a.remoteAgent != null) {
            int X1 = a.x;
            int Y1 = a.y;
            int X2 = a.remoteAgent.x;
            int Y2 = a.remoteAgent.y;
            g.setColor(AGENT_LINK_COLOR);
            Line2D.Double line = new Line2D.Double(X1, Y1, X2, Y2);
            g.draw(line);
          }
          g.setColor(AGENT_COLOR);
          g.drawString("udp" + String.valueOf(a.id), a.x - R + 2, a.y - R);
          if (this.sm.drawAgentDetail) g.drawString(Agent.convertType(a.agentType), a.x - R + 2, a.y - (3 *
R));
          break;
        case 3:
          g.setColor(AGENT_COLOR);
          g.drawString("null" + String.valueOf(a.id), a.x - R + 2, a.y - R);
          if (this.sm.drawAgentDetail) g.drawString(Agent.convertType(a.agentType), a.x - R + 2, a.y - (3 *
R)); case 5:
        case 6:
        case 7:
        case 9:
        case 10:
        case 11:
        case 13:
        case 14:
        case 15: } int X1 = a.x;

        int Y1 = a.y;
        int X2 = a.attachedNode.x;
        int Y2 = a.attachedNode.y;
        g.setColor(AGENT_COLOR);
        Line2D.Double line = new Line2D.Double(X1, Y1, X2, Y2);
        g.draw(line);
      }
    }

    private void drawApp(Graphics2D g)
    {
```

```
int R = 8;

int Rdiv2 = R / 2;

g.setStroke(new BasicStroke(1.0F, 1, 2));

String type = "";
Object[] apps = this.dm.getApps();
for (int i = 0; i < apps.length; ++i) {
  App a = (App)apps[i];
  g.setColor(APP_COLOR);
  Ellipse2D shap = new Ellipse2D.Double(a.x - Rdiv2, a.y - Rdiv2, R, R);
  g.fill(shap);
  switch (a.appType)
  {
  case 0:
    type = "ftp";
    break;
  case 1:
    type = "cbr";

  }

  g.drawString(type + String.valueOf(a.id), a.x - R + 2, a.y - R);
  int X1 = a.x;
  int Y1 = a.y;
  int X2 = a.agent.x;
  int Y2 = a.agent.y;
  Line2D.Double line = new Line2D.Double(X1, Y1, X2, Y2);
  g.draw(line);
 }
 }
}
```

SimulationParameterDialog.java

```
//The parameters dialog box for ssetting all the important parameters
package nsg;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.BufferedReader;
import java.io.BufferedWriter;
```

```
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.Writer;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTabbedPane;
import javax.swing.JTextField;

public class SimulationParamterDialog extends JDialog
 implements BasicParameters
{
 static final long serialVersionUID = 0L;
 JTextField simTimeItem = new JTextField("10.0");
 JTextField traceFileItem = new JTextField("out.tr");
 JTextField namFileItem = new JTextField("out.nam");
 JTabbedPane tab = new JTabbedPane();
 JButton done = new JButton("    Done    ");
 JButton save = new JButton("Save as default");
 JButton load = new JButton("Load");
 private void saveDefault()
 {
  String userHome = System.getProperty("user.home");
  File f = new File(userHome + "/nsg/");
  if (!(f.exists()))
   f.mkdirs();
  try
  {
   Writer out = new BufferedWriter(
    new FileWriter(userHome +
    "/nsg/wiredt.txt"));

   out.write(this.simTimeItem.getText() + " ");
   out.write(this.traceFileItem.getText() + " ");
   out.write(this.namFileItem.getText() + " ");




   out.flush();
   out.close();
```

```
  } catch (IOException e) {
    e.printStackTrace();
  }
 }

 private void loadDefault() {
  String userHome = System.getProperty("user.home");
  File f = new File(userHome + "/nsg/wirelessDefault.txt");
  if (!(f.exists()))
    return;
  try
  {
    BufferedReader in = new BufferedReader(
      new FileReader(userHome +
      "/nsg/wirelessDefault.txt"));
    String[] result = in.readLine().split(" ");
    this.simTimeItem.setText(result[0]);
    this.traceFileItem.setText(result[1]);
    this.namFileItem.setText(result[2]);
    in.close();
  } catch (IOException e) {
    e.printStackTrace();
  }
 }
 public SimulationParamterDialog(JFrame p)
 {
  super(p, true);
  loadDefault();

  setTitle("Simulation parameters setup");

  int w = Toolkit.getDefaultToolkit().getScreenSize().width;
  int h = Toolkit.getDefaultToolkit().getScreenSize().height;
  setBounds(w / 2 - 250, h / 2 - 300, 320, 200);

  JPanel panel = new JPanel();
  panel.setLayout(new FlowLayout());
  JPanel innerPanel = new JPanel();
  innerPanel.setLayout(new GridLayout(3, 2));
  innerPanel.add(new JLabel("Simulation time"));
  innerPanel.add(this.simTimeItem);
  innerPanel.add(new JLabel("Trace File"));
  innerPanel.add(this.traceFileItem);
  innerPanel.add(new JLabel("Nam File"));
  innerPanel.add(this.namFileItem);
  panel.add(innerPanel);
```

```
    this.tab.add("Simulation", panel);
    panel = new JPanel();
    panel.setLayout(new BorderLayout());

    panel = new JPanel();
    panel.add(this.done);
    this.done.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SimulationParamterDialog.this.setVisible(false);
      }
    });
    panel.add(this.save);
    this.save.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        SimulationParamterDialog.this.saveDefault();
      }
    });
    getContentPane().add(panel, "South");
    getContentPane().add(this.tab, "Center");
  }
}
```

BasicParameter.java

```
//Various parameters for setting of the tool
package nsg;

import java.awt.Color;

public abstract interface BasicParameters
{
  public static final long serialVersionUID = 0L;
  public static final Color DSEKTOP_BACKGROUND_COLOR = new Color(89, 222, 90);
  public static final Color INFO_COLOR = Color.PINK;
  public static final Color STATUS_LABEL_COLOR = new Color(56, 76, 229);
  public static final Color TOOLBAR_COLOR = new Color(212, 220, 202);
  public static final Color BUTTON_ENABLE_COLOR = Color.LIGHT_GRAY;
  public static final Color BUTTON_DISABLE_COLOR = new Color(238, 238, 238);
  public static final Color BACKGROUND_COLOR = Color.WHITE;

  public static final Color PANEL_COLOR = new Color(212, 220, 202);
  public static final Color NODE_COLOR = new Color(12, 122, 112);
  public static final Color NODE_TEXT_COLOR = Color.BLACK;
  public static final Color AGENT_COLOR = new Color(0, 128, 192);
  public static final Color AGENT_LINK_COLOR = Color.MAGENTA;
  public static final Color APP_COLOR = new Color(0, 187, 134);
  public static final Color IR_COLOR = Color.PINK;
  public static final Color SR_COLOR = new Color(168, 168, 168);
```

```
  public static final Color CONNECTED_COLOR = Color.BLUE;
  public static final Color LINK_COLOR = new Color(55, 54, 123);
  public static final Color TARGET_COLOR = Color.RED;
  public static final Color SRC_COLOR = Color.orange;
  public static final int WIRED_MODE = 1;
    public static final int NORMAL_MODE = 0;
  public static final int HAND_MODE = 1;
  public static final int CREATING_NODE_MODE = 2;
  public static final int CREATING_AGENT_MODE = 3;
  public static final int CREATING_APP_MODE = 4;
  public static final int CREATING_LINK_MODE = 5;
  public static final int AGENT_TCP = 0;
  public static final int AGENT_TCP_TAHOE = 4;
  public static final int AGENT_TCP_RENO = 8;
  public static final int AGENT_TCP_NEWRENO = 12;
  public static final int AGENT_TCP_VEGAS = 16;
  public static final int AGENT_TCP_SINK = 1;
  public static final int AGENT_UDP = 2;
  public static final int AGENT_NULL = 3;
  public static final int APP_FTP = 0;
  public static final int APP_CBR = 1;
  public static final int APP_PING = 2;
  public static final int APP_EXPONENTIAL = 3;
  public static final int APP_PARETO = 4;
  public static final int QUEUE_DROP_TAIL = 0;
  public static final int QUEUE_RED = 1;
  public static final int QUEUE_FQ = 2;
  public static final int QUEUE_DRR = 3;
  public static final int QUEUE_SFQ = 4;
  public static final int QUEUE_CBQ = 5;
  public static final int DUPLEX_LINK = 0;
  public static final int SIMPLEX_LINK = 1;
 }
```

DataStorage.java

```
package nsg;
//DATA storage of node ,link agent and application in to the array in form of Array LIST
 import java.util.ArrayList;
 import java.util.Iterator;
 import nsg.component.Agent;
 import nsg.component.App;
 import nsg.component.Link;
 import nsg.component.Node;
 public class DataStorage
 {
```

```
public ArrayList nodes;
public ArrayList links;
public ArrayList agents;
public ArrayList apps;
public static Iterator it;
public DataStorage()
{
  this.nodes = new ArrayList();
  this.links = new ArrayList();
  this.agents = new ArrayList();
  this.apps = new ArrayList();
 }
public void removeApp(App app) {
  this.apps.remove(app);
}
public void removeAgent(Agent agent)
{
  this.agents.remove(agent);

  checkAgents();
  checkApps();
}
 public void removeLink(Link link)
 {
  this.links.remove(link);
 }
 public void checkApps() {
  Object[] apps = getApps();

  for (int i = 0; i < apps.length; ++i) {
    App app = (App)apps[i];
    if (!(this.agents.contains(app.agent)))
      this.apps.remove(app);
   }
 }
 public void checkAgents()
 {
  Agent agent;
  Object[] agents = getAgents();

  for (int i = 0; i < agents.length; ++i) {
    agent = (Agent)agents[i];
    if (!(this.nodes.contains(agent.attachedNode))) {
      this.agents.remove(agent);
     }
   }
```

```
 for (int i = 0; i < agents.length; ++i) {
  agent = (Agent)agents[i];
  if (!(this.agents.contains(agent.remoteAgent)))
   agent.remoteAgent = null;
 }
}
public void checkLinks()
{
 Object[] links = getLinks();

 for (int i = 0; i < links.length; ++i) {
  Link link = (Link)links[i];
  if ((!(this.nodes.contains(link.src))) || (!(this.nodes.contains(link.dst))))
   this.links.remove(link);
 }
}
public void removeNode(Node node)
{
 this.nodes.remove(node);
 checkLinks();
 checkAgents();
 checkApps();
}



public Node findNode(int x, int y)
{
 it = this.nodes.iterator();

 while (it.hasNext()) {
  Node p = (Node)it.next();
  if ((Math.abs(p.x - x) < 10) && (Math.abs(p.y - y) < 10)) {
   return p;
  }
 }
 return null;
}
 public Link findLink(int x, int y) {
 it = this.links.iterator();

 Link tempP = null;
 double tempD = 0.0D;
 while (it.hasNext()) {
  Link linkP = (Link)it.next();
  boolean temp = ((linkP.src.x > x) ? true : false) ^ ((linkP.dst.x > x) ? true : false);
```

```
    if (!(temp)) {
      continue;
    }
    temp = ((linkP.src.y > y) ? true : false) ^ ((linkP.dst.y > y) ? true : false);
    if (!(temp)) {
      continue;
    }
    int x1 = linkP.src.x;
    int x2 = linkP.dst.x;
    int y1 = linkP.src.y;
    int y2 = linkP.dst.y;
    double d = Math.abs((y2 - y1) * x + (x1 - x2) * y + y1 * (x2 - x1) + x1 * (y1 - y2)) / Math.sqrt((y2 -
y1) * (y2 - y1) + (x1 - x2) * (x1 - x2));
    if (d < 40.0D) {
      if (tempP == null) {
        tempP = linkP;
        tempD = d;
      } else if (tempD > d) {
        tempP = linkP;
        tempD = d;
      }
    }
  }
  return tempP;
}
public Agent findAgent(int x, int y) {
  it = this.agents.iterator();
  while (it.hasNext()) {
    Agent p = (Agent)it.next();
    if ((Math.abs(p.x - x) < 10) && (Math.abs(p.y - y) < 10)) {
      return p;
    }
  }
  return null;
}

public App findApp(int x, int y) {
  it = this.apps.iterator();

  while (it.hasNext()) {
    App p = (App)it.next();
    if ((Math.abs(p.x - x) < 10) && (Math.abs(p.y - y) < 10)) {
      return p;
    }
  }
  return null;
```

```
  }

  public Object[] getNodes() {
   return this.nodes.toArray();
  }

  public Object[] getLinks() {
   return this.links.toArray();
  }

  public Object[] getAgents() {
   return this.agents.toArray();
  }

  public Object[] getApps() {
   return this.apps.toArray();
  }
}
```

TCLGenerator.java

```
// This code is used to generate the TCL script


public class TCLGenerator extends JInternalFrame
 implements BasicParameters
{
 static final long serialVersionUID = 0L;
 JTextPane tclArea = new JTextPane();
 File tclFile;
 JFileChooser tcljfc = new JFileChooser();
 SimpleAttributeSet titleAttr = new SimpleAttributeSet();
 SimpleAttributeSet tclAttr = new SimpleAttributeSet();
 SimpleAttributeSet noteAttr = new SimpleAttributeSet();
 Document doc;
 JToolBar toolbar = new JToolBar();
 ScenarioFrame sm;
 SimulationParamterDialog pm;
 DataStorage dm;

 public TCLGenerator(ScenarioFrame sm)
 {
  super("WiredTcl", true, true, true, true);
  this.sm = sm;
  this.dm = sm.sv.dm;
  this.pm = sm.parameters;

  this.tcljfc.addChoosableFileFilter(new FileFilter() {
```

```
  public boolean accept(File file) {
   if (file.isDirectory()) {
    return true;
   }
   return (file.getName().endsWith(".tcl"));
  }

  public String getDescription()
  {
   return "TCL files (*.tcl)";
  }
});
StyleConstants.setFontSize(this.titleAttr, 14);
StyleConstants.setFontFamily(this.titleAttr, "Courier New");
StyleConstants.setBold(this.titleAttr, true);
StyleConstants.setForeground(this.titleAttr, Color.BLUE);

StyleConstants.setFontSize(this.tclAttr, 14);
StyleConstants.setFontFamily(this.tclAttr, "Courier New");
StyleConstants.setBold(this.tclAttr, false);
StyleConstants.setForeground(this.tclAttr, Color.BLACK);

StyleConstants.setFontSize(this.noteAttr, 14);
StyleConstants.setFontFamily(this.noteAttr, "Courier New");
StyleConstants.setBold(this.noteAttr, true);
StyleConstants.setForeground(this.noteAttr, new Color(32, 158, 29));
this.doc = this.tclArea.getDocument();

JButton b = new JButton(" Save ");
b.setMnemonic('S');
b.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  TCLGenerator.this.saveTclFile();
 }
});
this.toolbar.add(b);
b = new JButton(" Save as ");
b.setMnemonic('A');
b.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  TCLGenerator.this.saveTclFileAs();
 }
});
this.toolbar.add(b);
this.toolbar.setFloatable(false);
```

```
 getContentPane().add(this.toolbar, "North");
 getContentPane().add(new JScrollPane(this.tclArea), "Center");
 generate();
}

private void generate() {
 appendTitle("# This script is generated by automated Tool\n");
 appendTitle("# This can be run with help of network simulator2\n");

 parametersSetup();
 initialization();
 nodesDefinition();
 linksDefinition();
 agentsDefinition();
 appDefinition();
 termination();
 }

// the TCL script having different segment have different code
 private void parametersSetup() {
  appendNote("\n#==========================================\n");
  appendNote("#    Simulation parameters setup\n");
  appendNote("#==========================================\n");

   appendTCL("set val(stop)   " + this.pm.simTimeItem.getText() + "                    ", ";# total time of
runnning simulation\n");
  }

  private void initialization() {
   boolean enableTrace = !(this.pm.traceFileItem.getText().equals(""));
   boolean enableNAM = !(this.pm.namFileItem.getText().equals(""));

   appendNote("\n#==========================================\n");
   appendNote("#      Initialization      \n");
   appendNote("#==========================================\n");


    }

   }
   private void nodesDefinition() {
    appendNote("\n#==========================================\n");
    appendNote("#      Nodes Definition      \n");
    appendNote("#==========================================\n");

    }
```

```
private void linksDefinition()
{
 Link link;
 appendNote("\n#=====================================\n");
 appendNote("#       Links Definition       \n");
 appendNote("#=====================================\n");


}

private void agentsDefinition()
{
 appendNote("\n#=====================================\n");
 appendNote("#       Agents Definition       \n");
 appendNote("#=====================================\n");


}

private void appDefinition()
{
 appendNote("\n#=====================================\n");
 appendNote("#       Applications Definition       \n");
 appendNote("#=====================================\n");

}

private void termination()
{
 boolean enableTrace = !(this.pm.traceFileItem.getText().equals(""));
 boolean enableNAM = !(this.pm.namFileItem.getText().equals(""));

 appendNote("\n#=====================================\n");
 appendNote("#       Termination       \n");
 appendNote("#=====================================\n");
 appendNote("#Define a 'finish' procedure\n");




}
//All these append function use different format of writing into the file
 public void appendTitle(String s) {
  try {
```

```
    this.doc.insertString(this.tclArea.getCaretPosition(), s, this.titleAttr);
   } catch (Exception evt) {
    evt.printStackTrace();
   }
  }

  public void appendNote(String note) {
   try {
    this.doc.insertString(this.tclArea.getCaretPosition(), note, this.noteAttr);
   } catch (Exception evt) {
    evt.printStackTrace();
   }
  }

  public void appendTCL(String tcl) {
   try {
    this.doc.insertString(this.tclArea.getCaretPosition(), tcl, this.tclAttr);
   } catch (Exception evt) {
    evt.printStackTrace();
   }
  }

  public void appendTCL(String tcl, String note) {
   try {
    this.doc.insertString(this.tclArea.getCaretPosition(), tcl, this.tclAttr);
    this.doc.insertString(this.tclArea.getCaretPosition(), note, this.noteAttr);
   } catch (Exception evt) {
    evt.printStackTrace();
   }
  }

  public boolean saveTclFileAs() {
   try {
    this.tcljfc.setDialogTitle("Please select file");
    int m = this.tcljfc.showSaveDialog(this);
    if (m == 0) {
     this.tclFile = this.tcljfc.getSelectedFile();
     if (!(this.tclFile.getAbsolutePath().endsWith("tcl"))) {
      this.tclFile = new File(this.tclFile.getAbsolutePath() + ".tcl");
     }
     saveTclFile();
     return true;
    }
    return false;
   }
   catch (Exception evt) {
```

```
    System.out.println(evt.getMessage()); }
   return false;
  }

  public boolean saveTclFile()
  {
   try {
    if ((this.tclFile == null) &&
     (!(saveTclFileAs()))) {
     return false;
    }

    Writer out = new OutputStreamWriter(new FileOutputStream(this.tclFile));

    out.write(this.tclArea.getText());
    out.flush();
    out.close();
    setTitle(this.tclFile.getAbsolutePath());
    return true;
   } catch (Exception evt) {
    System.out.println(evt.getMessage()); }
   return false;
  }
 }
```

DelayMenuComparison.java

```
public class DelayMenuComparison extends JFrame{

// This code is used to create delay graph algorithm is based on Awk code .
// Similary all the performance metrics are generated


   public DelayMenuComparison()
   {
      int w = Toolkit.getDefaultToolkit().getScreenSize().width;
      int h = Toolkit.getDefaultToolkit().getScreenSize().height;
      this.setBounds(w / 2 - 250, h / 2 - 300,620, 90);
      this.dataset=new XYSeriesCollection();
      this.setTitle("Tell trace file for analysis");
      this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JPanel panel1 =new JPanel();

      JLabel rev =new JLabel("Delay Diagram tel the total time taken for a packets to reach to receiver to
sender.");
```

```
    panel1.add(rev);
    open =new JButton("Open");
    open.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
       DelayMenuComparison.this.dispose();
      DelayMenuComparison.this.evtPerformed();
      }
    });

    panel1.add(open);
    this.add(panel1);
    this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    this.setVisible(true);


  }

  public void evtPerformed()
  {

    JFileChooser tracejfc=new JFileChooser();
          try
           {
             tracejfc.setDialogTitle("Please Select File");
              tracejfc.setMultiSelectionEnabled(true);
              int m = tracejfc.showOpenDialog(null);
              if(m == 0)
              {
                 File traceFile[]= tracejfc.getSelectedFiles();
                 for(int ik=0;ik<traceFile.length;ik++)
                 {
                    series =traceFile[ik].toString();
                    int l=0;
                    l= series.length();
                    series=series.substring(l-3);
                    if(!series.equals(".tr"))
                    {
                       JOptionPane.showMessageDialog(DelayMenuComparison.this,"Not atrace
File","Error!!",JOptionPane.INFORMATION_MESSAGE);
                    }
                    else
                    {
                       this.dispose();
                       startTime=new double[200000];
                       endTime=new double[200000];
                       higestpktID=0;
```

```
                    analysisFile(traceFile[ik]);
                    series=traceFile[ik].toString();
                    getseries(series);
                    dataset.addSeries(series1);

                  }
                }
                e1 =new GraphDrawComparison("Delay Diagram",dataset,"Delay Diagram for Given
Scenario","Packet Id","Time(in seconds)");
                  e1.pack();

                  RefineryUtilities.centerFrameOnScreen(e1);
                   e1.dispose();
                   e1.setVisible(true);

              } else
              {
                 return;
              }
            }
            catch(Exception evt)
            {
               System.out.println(evt.getMessage());
               return;
            }
  }


  private void analysisLine(byte line[])
   {
     Vector v = new Vector();
     String tmp = "";
     for(int i = 0; i < line.length; i++)
        if(line[i] == 32)
        {
          if(!tmp.equals(""))
          {
            v.add(tmp);
            tmp = "";
          }
        } else
        {
          tmp = (new
StringBuilder(String.valueOf(tmp))).append(String.valueOf((char)line[i])).toString();
        }
       v.add(tmp);
```

```
String token[] = new String[v.size()];
for(int i = 0; i < v.size(); i++)
   token[i] = (String)v.get(i);

TraceItem item = new TraceItem();
item.event=token[0];
item.time=Double.parseDouble(token[1]);
item.from=Integer.parseInt(token[2]);
item.to=Integer.parseInt(token[3]);
item.src=(int)Double.parseDouble(token[8]);
item.dst=(int)Double.parseDouble(token[9]);
item.packetId=Integer.parseInt(token[11]);

 if ( item.packetId > higestpktID )
  higestpktID = item.packetId;


 if ( startTime[item.packetId] == 0 )
  startTime[item.packetId] = item.time;


  if ( ((item.event).equals("r")) && (item.to==item.dst)) {
    endTime[item.packetId] = item.time;
  } else {
   endTime[item.packetId] = -1;
   }


 }
private void analysisFile(File f)
 {

    try
    {
       BufferedReader in = new BufferedReader(new InputStreamReader(new FileInputStream(f)));

       for(String line = in.readLine(); line != null; line = in.readLine())
       {
          //System.out.println(line);
          analysisLine(line.getBytes());
       }
    }
    catch(Exception e)
    {
       System.out.println(e.getMessage());
    }
```

```
  }
 private void getseries(String s1) {
    series1=new XYSeries(s1);
    int pkt1;
    double pktDuration;
    for ( pkt1 = 0; pkt1 <= higestpktID; pkt1++ ) {
        pktDuration = endTime[pkt1] - startTime[pkt1];
        if ( startTime[pkt1] < endTime[pkt1] ) series1.add(pkt1,pktDuration);
     }

   }
}
```

GraphDraw.java

```
// This code is used to draw the performance metrics based on trace file

      public class GraphDraw  extends ApplicationFrame {
         public JFreeChart chart;
         public JMenuBar menubar = new JMenuBar();
         public JButton b1;
         public GraphDraw( String title,XYSeries series,String ch1,String ch2,String ch3) {
            super(title);
            XYDataset xyDataset = new XYSeriesCollection(series);
            chart= ChartFactory.createXYLineChart(ch1, ch2, ch3,
                  xyDataset, PlotOrientation.VERTICAL, true, true, false);
            XYPlot plot = chart.getXYPlot();
            chart.setBackgroundPaint(new GradientPaint(0,500, Color.white, 1000, 300, Color.blue));
            plot.setBackgroundPaint(new GradientPaint(0,0, Color.white, 1000, 0, Color.green));


            plot.setDomainCrosshairLockedOnData(true);
            plot.setDomainCrosshairVisible(true);
            plot.setRangeCrosshairLockedOnData(true);
            plot.setRangeCrosshairVisible(true);

            ChartPanel chartPanel = new ChartPanel(chart);
            chartPanel.setPreferredSize(new java.awt.Dimension(1100, 870));
            chartPanel.setMouseWheelEnabled(true);
            getContentPane().add(chartPanel,"South");
            b1 =new JButton("Save");
            setJMenuBar(this.menubar);
            JMenu menu = new JMenu("File");
            menu.setMnemonic('F');
```

```
        JMenuItem item = new JMenuItem("Save");
        item.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            try {
              JFileChooser tracejfc=new JFileChooser();
              tracejfc.setDialogTitle("Save the diagram");
              int retVal = tracejfc.showSaveDialog(null);
              if (retVal == JFileChooser.APPROVE_OPTION) {
                File thefile = tracejfc.getSelectedFile();
                if (!(thefile.getAbsolutePath().endsWith("jpg"))) {
                  thefile = new File(thefile.getAbsolutePath() + ".jpg");
                }
                ChartUtilities.saveChartAsJPEG(thefile, chart, 1100, 870);
              }
            }catch (Exception e1) {
              System.out.println("Problem occurred creating chart.");
            }
          }
        });
        item.setMnemonic('S');
        menu.add(item);
        this.menubar.add(menu);
        item = new JMenuItem("Close");
        item.setMnemonic('C');
        item.addActionListener(new ActionListener() {
          public void actionPerformed(ActionEvent e) {
            GraphDraw.this.dispose();
          }
        });
        menu.add(item);
        this.menubar.add(menu);

    }
}
```

GraphDrawComparison.java

```
// This code is used to draw comparison graph of the algorithm
public class GraphDrawComparison extends ApplicationFrame {

public String ch1,ch2,ch3;
public JMenuBar menubar = new JMenuBar();
    public JButton b1;
```

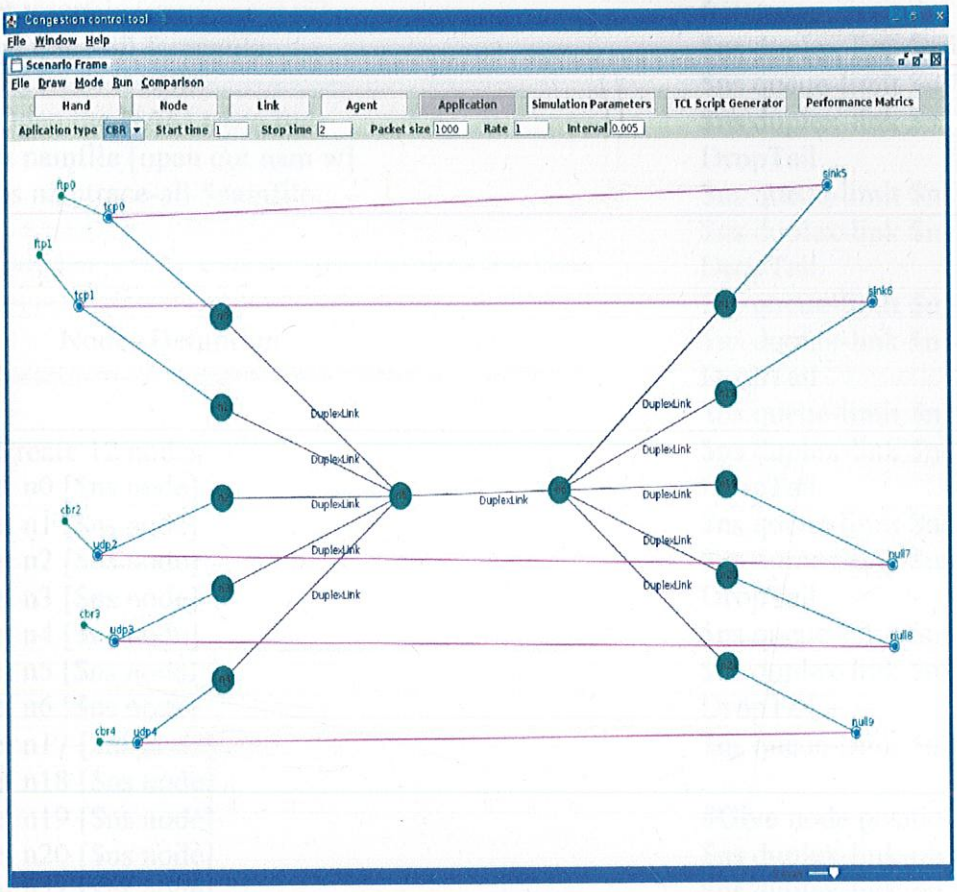
```
public GraphDrawComparison( String title,XYSeriesCollection cdset,String chk1,String chk2,String
chk3) {
    super(title);
    ch1=chk1;
    ch2=chk2;
    ch3=chk3;
    final XYSeriesCollection dataset = cdset;
    final JFreeChart chart = createChart(dataset);

    final ChartPanel chartPanel = new ChartPanel(chart);


    chartPanel.setPreferredSize(new java.awt.Dimension(1100, 870));
    chartPanel.setMouseWheelEnabled(true);


    getContentPane().add(chartPanel,"South");
        b1 =new JButton("Save");
    setJMenuBar(this.menubar);

    JMenu menu = new JMenu("File");
    menu.setMnemonic('F');

    JMenuItem item = new JMenuItem("Save");
    item.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        try {
            chartPanel.doSaveAs();
      }catch (Exception e1) {
          System.out.println("Problem occurred creating chart.");
      }
      }
    });
    item.setMnemonic('S');
    menu.add(item);
    this.menubar.add(menu);
      item = new JMenuItem("Close");
    item.setMnemonic('C');
    item.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        GraphDrawComparison.this.dispose();
      }
    });
    menu.add(item);
    this.menubar.add(menu);
    this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
```

```
}


private JFreeChart createChart(XYSeriesCollection dataset) {

    final JFreeChart chart = ChartFactory.createXYLineChart(
        ch1,      // chart title
        ch2,                // domain axis label
        ch3,               // range axis label
        dataset,                // data
        PlotOrientation.VERTICAL, // orientation
        true,                // include legend
        true,                // tooltips
        false                // urls
    );

    final XYPlot plot = chart.getXYPlot();
    plot.setBackgroundPaint(Color.lightGray);
    plot.setDomainGridlinePaint(Color.white);
    plot.setRangeGridlinePaint(Color.white);


    chart.setBackgroundPaint(new GradientPaint(0,500, Color.white, 1000, 300, Color.blue));

    plot.setDomainCrosshairLockedOnData(true);
    plot.setDomainCrosshairVisible(true);
    plot.setRangeCrosshairLockedOnData(true);
    plot.setRangeCrosshairVisible(true);

    final XYLineAndShapeRenderer renderer =  (XYLineAndShapeRenderer) plot.getRenderer();
    renderer.setShapesFilled(true);
    renderer.setItemLabelsVisible(true);
    final ItemLabelPosition p = new ItemLabelPosition(
       ItemLabelAnchor.OUTSIDE12, TextAnchor.BOTTOM_CENTER, TextAnchor.CENTER,
Math.PI / 4
    );
    renderer.setPositiveItemLabelPosition(p);

    // change the auto tick unit selection to integer units only...
    final NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
    rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());
    // OPTIONAL CUSTOMISATION COMPLETED.
    return chart;
}}
```

Chapter 8 Testing

8.1 Test Case

Simulation Scenario



TCL script

```
# This script is generated by automated Tool
# This can be run with help of network
simulator2

#===================================
==
#     Simulation parameters setup
#===================================
==
set val(stop)   50.0                ;# total time
of runnning simulation

#===================================
==
#       Initialization
#===================================
==
#Create a ns simulator
set ns [new Simulator]

$ns color 1 Blue
$ns color 2 Red
$ns color 3 White
$ns color 4 Brown
$ns color 5 Black
```

```
$ns color 6 Purple
$ns color 7 Orange
$ns color 8 Yellow
$ns color 9 Pink
$ns color 10 Green
#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

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

#=====================================
==
#       Nodes Definition
#=====================================
==
#Create 12 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n17 [$ns node]
set n18 [$ns node]
set n19 [$ns node]
set n20 [$ns node]
set n21 [$ns node]

#=====================================
==
#       Links Definition
#=====================================
==
#Createlinks between nodes
$ns duplex-link $n0 $n5 100.0Mb 10ms
DropTail
$ns queue-limit $n0 $n5 50
$ns duplex-link $n1 $n5 100.0Mb 10ms
DropTail
$ns queue-limit $n1 $n5 50
$ns duplex-link $n5 $n3 100.0Mb 10ms
DropTail
$ns queue-limit $n5 $n3 50
$ns duplex-link $n4 $n5 100.0Mb 10ms
DropTail
$ns queue-limit $n4 $n5 50
$ns duplex-link $n2 $n5 100.0Mb 10ms
DropTail
$ns queue-limit $n2 $n5 50
$ns duplex-link $n5 $n6 50.0Mb 40ms RED
$ns queue-limit $n5 $n6 20
$ns duplex-link $n17 $n6 100.0Mb 10ms
DropTail
$ns queue-limit $n17 $n6 50
$ns duplex-link $n17 $n6 100.0Mb 10ms
DropTail
$ns queue-limit $n17 $n6 50
$ns duplex-link $n18 $n6 100.0Mb 10ms
DropTail
$ns queue-limit $n18 $n6 50
$ns duplex-link $n6 $n19 100.0Mb 10ms
DropTail
$ns queue-limit $n6 $n19 50
$ns duplex-link $n6 $n20 100.0Mb 10ms
DropTail
$ns queue-limit $n6 $n20 50
$ns duplex-link $n21 $n6 100.0Mb 10ms
DropTail
$ns queue-limit $n21 $n6 50

#Give node position (for NAM)
$ns duplex-link-op $n0 $n5 orient right-down
$ns duplex-link-op $n1 $n5 orient right-down
$ns duplex-link-op $n5 $n3 orient left-down
$ns duplex-link-op $n4 $n5 orient right-up
$ns duplex-link-op $n2 $n5 orient right
$ns duplex-link-op $n5 $n6 orient right
$ns duplex-link-op $n17 $n6 orient left-down
$ns duplex-link-op $n17 $n6 orient left-down
$ns duplex-link-op $n18 $n6 orient left-down
$ns duplex-link-op $n6 $n19 orient right
$ns duplex-link-op $n6 $n20 orient right-down
$ns duplex-link-op $n21 $n6 orient left-up

#=====================================
==
#       Agents Definition
```

```
#=====================================
==
#Setup a TCP connection
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink5 [new Agent/TCPSink]
$ns attach-agent $n17 $sink5
$ns connect $tcp0 $sink5
$tcp0 set packetSize_ 2000
$tcp0 set window_ 8000
$tcp0 set fid_ 1

#Setup a TCP/FullTcp/Tahoe connection
set tcp1 [new Agent/TCP/FullTcp/Tahoe]
$ns attach-agent $n1 $tcp1
set sink6 [new Agent/TCPSink]
$ns attach-agent $n18 $sink6
$ns connect $tcp1 $sink6
$tcp0 set window_ 8000
$tcp1 set packetSize_ 2000

$tcp1 set fid_ 2
#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set null7 [new Agent/Null]
$ns attach-agent $n19 $null7
$ns connect $udp2 $null7
$udp2 set packetSize_ 2000

$udp2 set fid_ 3
#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n3 $udp3
set null8 [new Agent/Null]
$ns attach-agent $n20 $null8
$ns connect $udp3 $null8
$udp3 set packetSize_ 1500

$udp3 set fid_ 4
#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n4 $udp4
set null9 [new Agent/Null]
$ns attach-agent $n21 $null9
$ns connect $udp4 $null9

$udp4 set packetSize_ 1500

$udp4 set fid_ 5

#=====================================
==
#       Applications Definition
#=====================================
==
#Setup a FTP Application over TCP connection
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set type_ FTP
$ns at 2.0 "$ftp0 start"
$ns at 49.0 "$ftp0 stop"

#Setup a FTP Application over
TCP/FullTcp/Tahoe connection
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp0 set type_ FTP
$ns at 8.0 "$ftp1 start"
$ns at 44.0 "$ftp1 stop"

#Setup a CBR Application over UDP
connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 1.0Mb
$cbr2 set random_
$ns at 13.0 "$cbr2 start"
$ns at 40.0 "$cbr2 stop"

#Setup a CBR Application over UDP
connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp3
$cbr3 set packetSize_ 1000
$cbr3 set rate_ 1.0Mb
$cbr3 set random_
$ns at 18.0 "$cbr3 start"
$ns at 35.0 "$cbr3 stop"

#Setup a CBR Application over UDP
connection
```

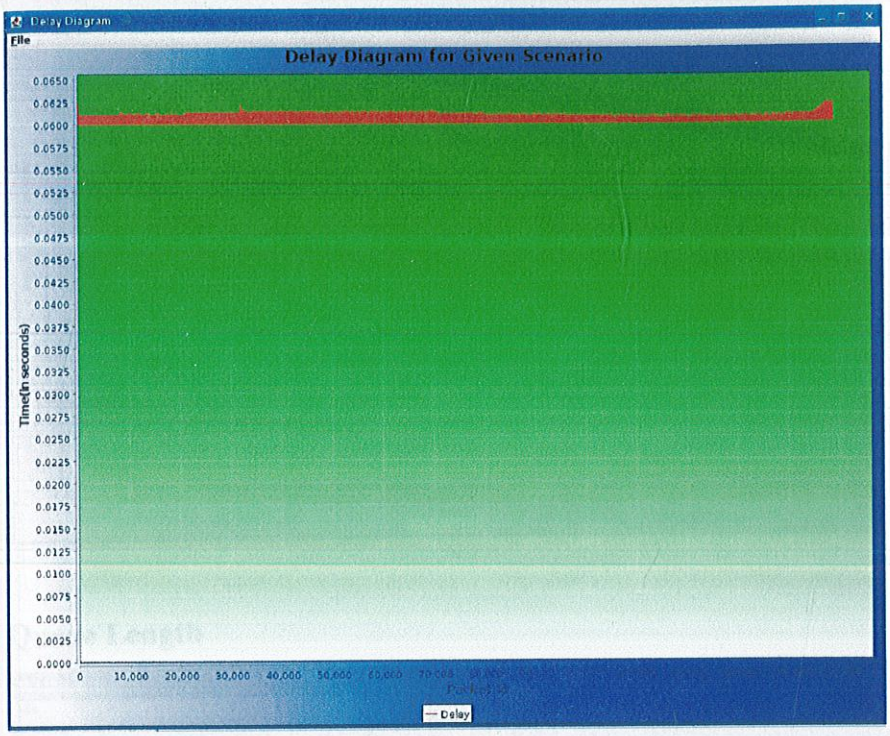
```
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 1.0Mb
$cbr4 set random_
$ns at 23.0 "$cbr4 start"
$ns at 33.0 "$cbr4 stop"


#=====================================
==
#       Termination
#=====================================
==
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
$ns at $val(stop) "finish"
$ns run
```

Sample trace file Generated:

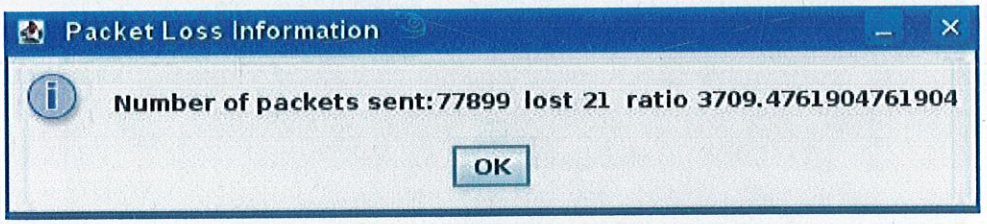
```
+ 2 0 5 tcp 40 ------- 1 0.0 7.0 0 0
- 2 0 5 tcp 40 ------- 1 0.0 7.0 0 0
r 2.010003 0 5 tcp 40 ------- 1 0.0 7.0 0 0
+ 2.010003 5 6 tcp 40 ------- 1 0.0 7.0 0 0
- 2.010003 5 6 tcp 40 ------- 1 0.0 7.0 0 0
r 2.05001 5 6 tcp 40 ------- 1 0.0 7.0 0 0
+ 2.05001 6 7 tcp 40 ------- 1 0.0 7.0 0 0
- 2.05001 6 7 tcp 40 ------- 1 0.0 7.0 0 0
r 2.060013 6 7 tcp 40 ------- 1 0.0 7.0 0 0
+ 2.060013 7 6 ack 40 ------- 1 7.0 0.0 0 1
- 2.060013 7 6 ack 40 ------- 1 7.0 0.0 0 1
r 2.070016 7 6 ack 40 ------- 1 7.0 0.0 0 1
+ 2.070016 6 5 ack 40 ------- 1 7.0 0.0 0 1
- 2.070016 6 5 ack 40 ------- 1 7.0 0.0 0 1
r 2.110022 6 5 ack 40 ------- 1 7.0 0.0 0 1
+ 2.110022 5 0 ack 40 ------- 1 7.0 0.0 0 1
- 2.110022 5 0 ack 40 ------- 1 7.0 0.0 0 1
r 2.120026 5 0 ack 40 ------- 1 7.0 0.0 0 1
+ 2.120026 0 5 tcp 2040 ------- 1 0.0 7.0 1 2
+ 49.062054 5 0 ack 40 ------- 1 7.0 0.0 71098 149016
- 49.062054 5 0 ack 40 ------- 1 7.0 0.0 71098 149016
r 49.062381 6 5 ack 40 ------- 1 7.0 0.0 71099 149017
+ 49.062381 5 0 ack 40 ------- 1 7.0 0.0 71099 149017
- 49.062381 5 0 ack 40 ------- 1 7.0 0.0 71099 149017
r 49.062707 6 5 ack 40 ------- 1 7.0 0.0 71100 149018
+ 49.062707 5 0 ack 40 ------- 1 7.0 0.0 71100 149018
- 49.062707 5 0 ack 40 ------- 1 7.0 0.0 71100 149018
r 49.063034 6 5 ack 40 ------- 1 7.0 0.0 71101 149019
+ 49.063034 5 0 ack 40 ------- 1 7.0 0.0 71101 149019
- 49.063034 5 0 ack 40 ------- 1 7.0 0.0 71101 149019
```

Performance Metric

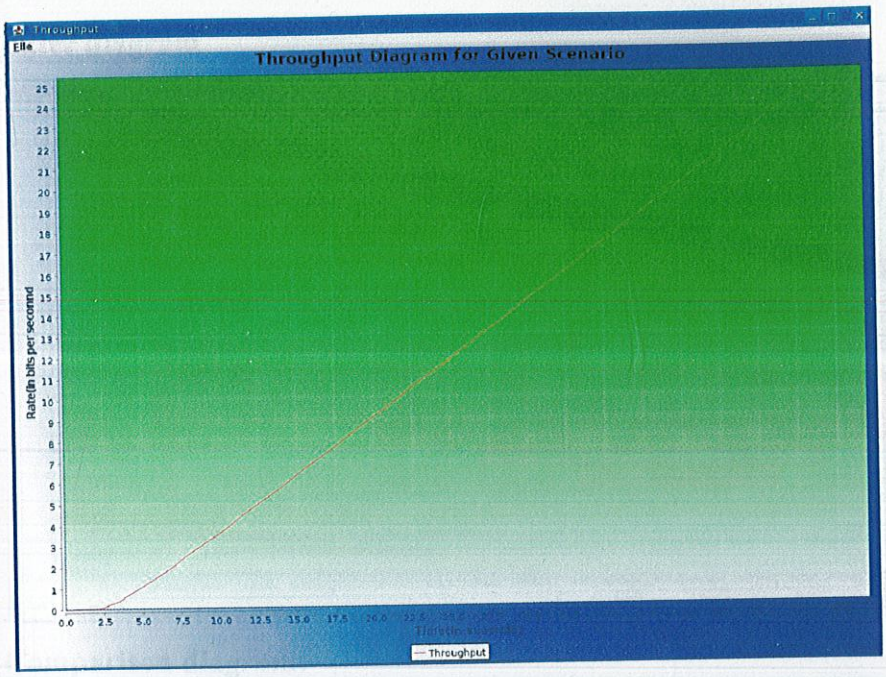
a) Delay



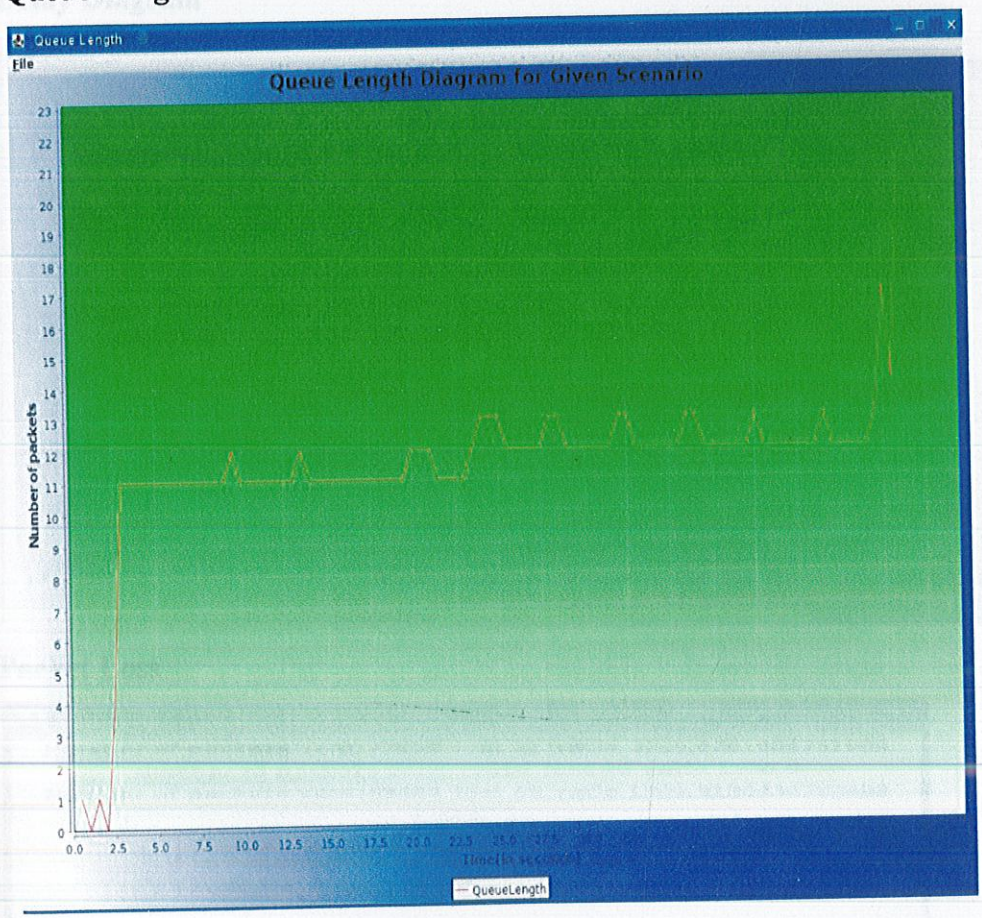
b) Packet Loss



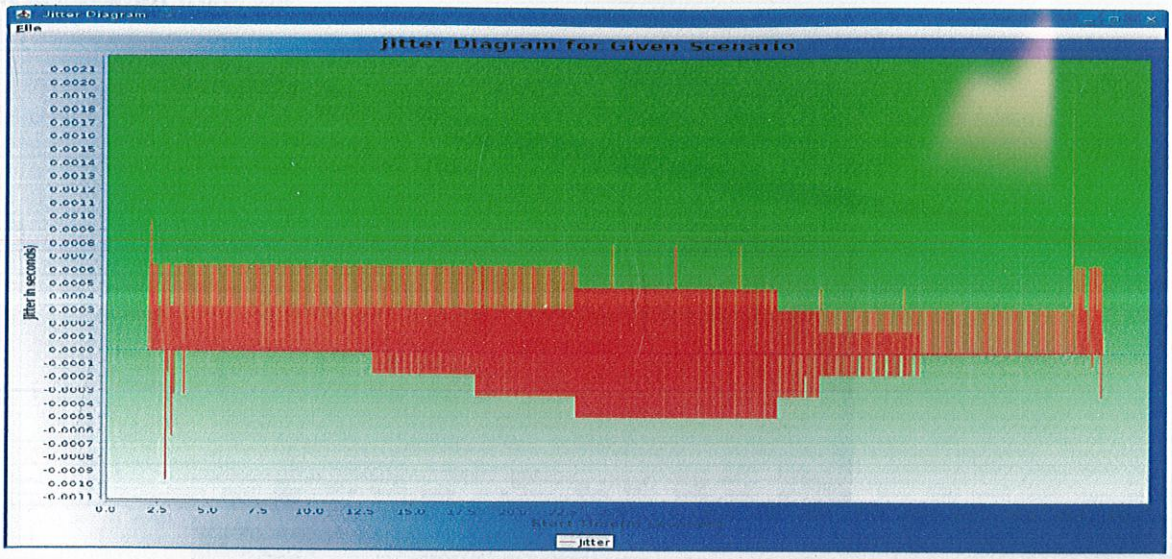
c) Throughput Diagram



d) **Queue Length**



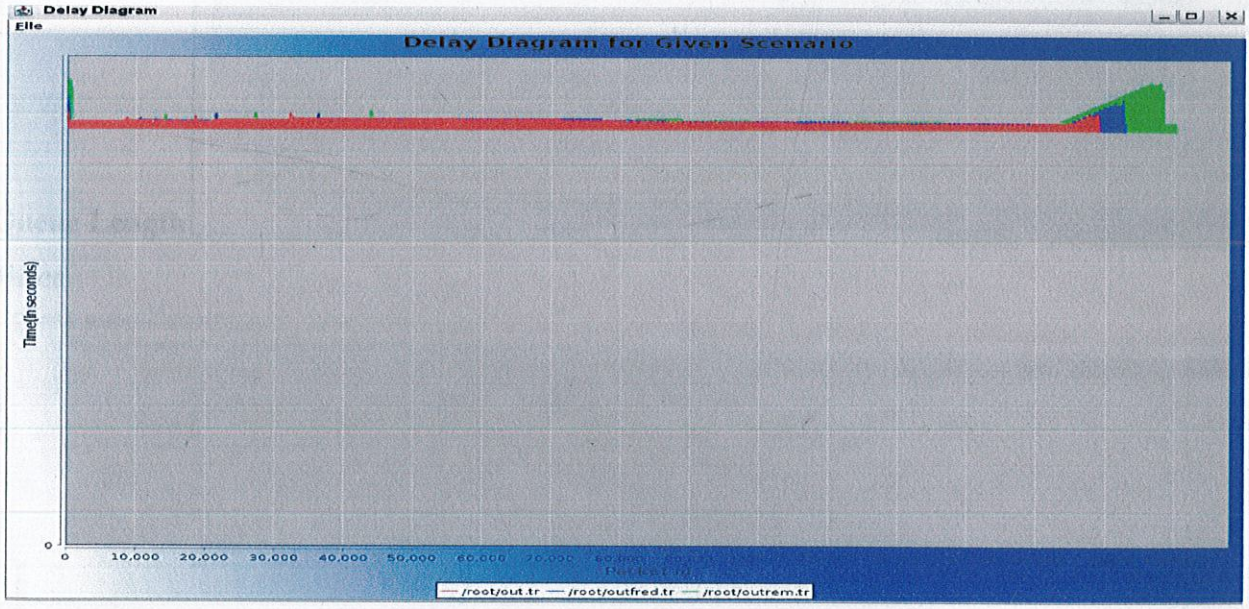
e) **Jitter diagram**



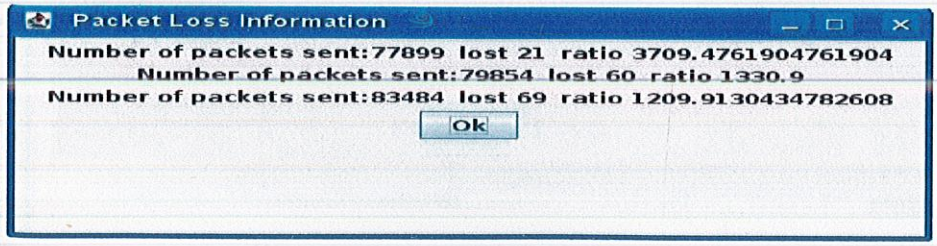
Comparison diagram:

Same simulation is done with REM and FRED algorithm using tool

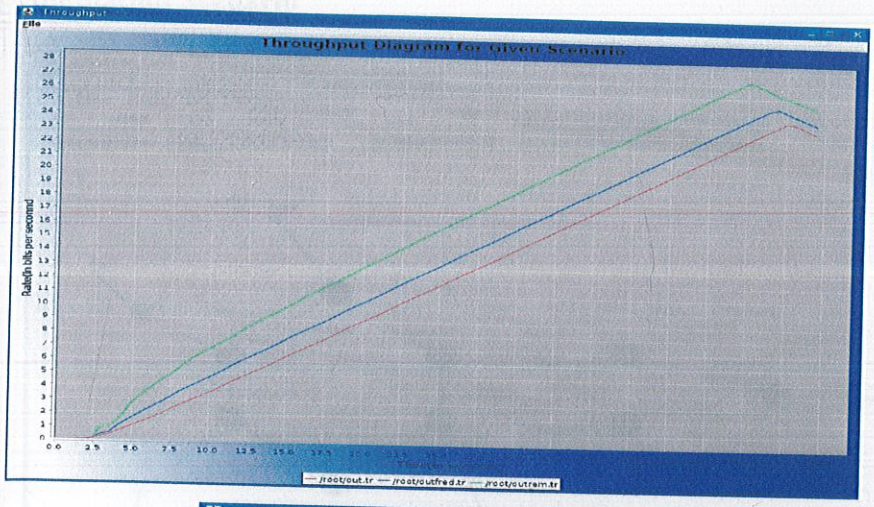
) **Delay Diagram**

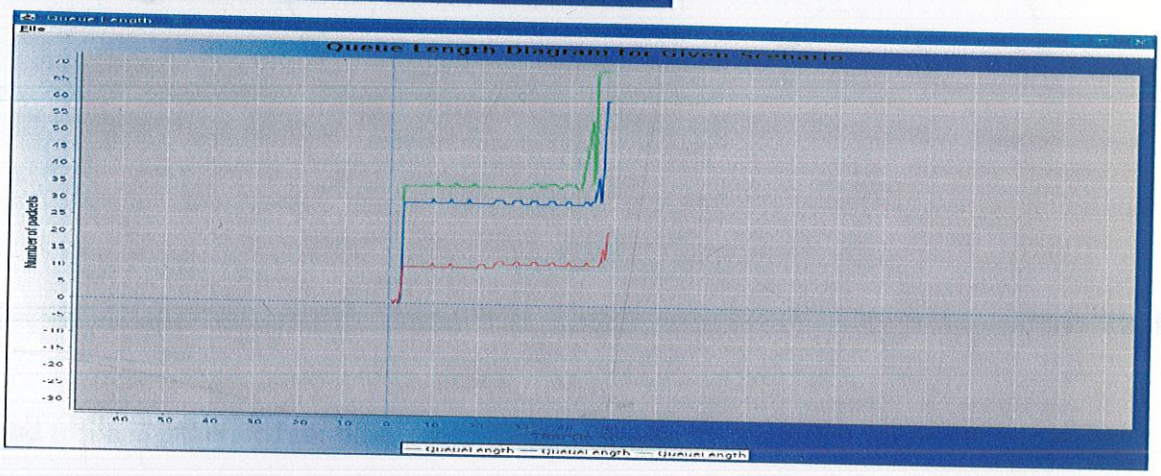


) **Packet Loss**



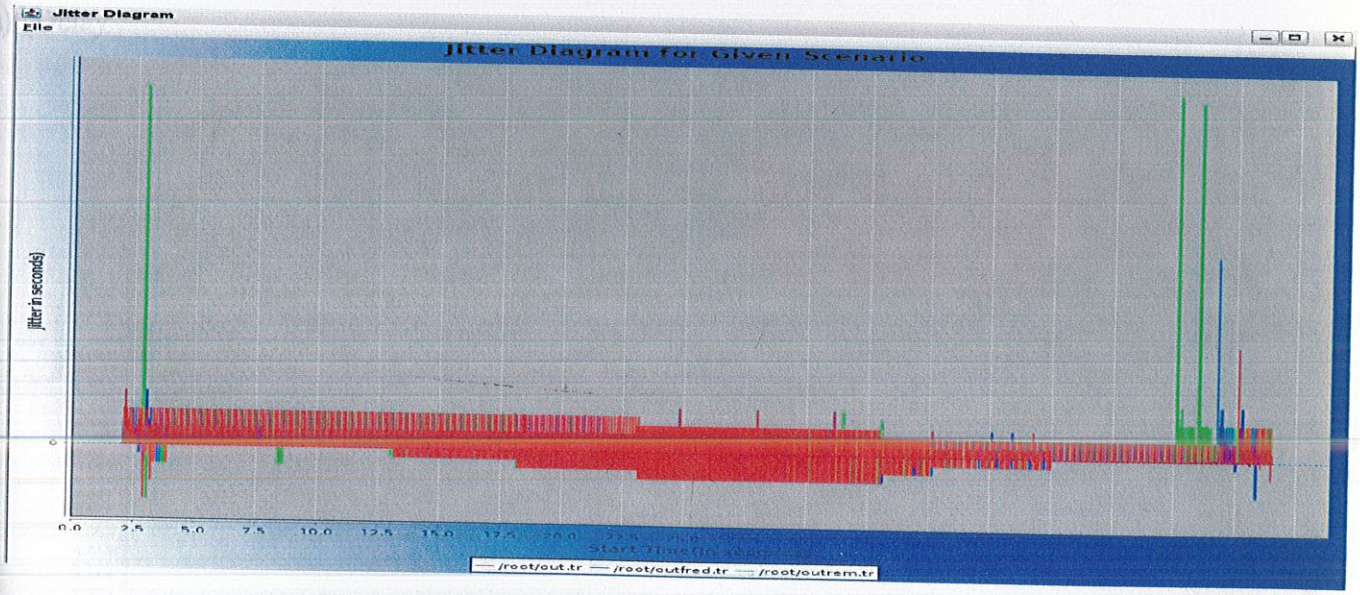
c) **Throughput diagram**





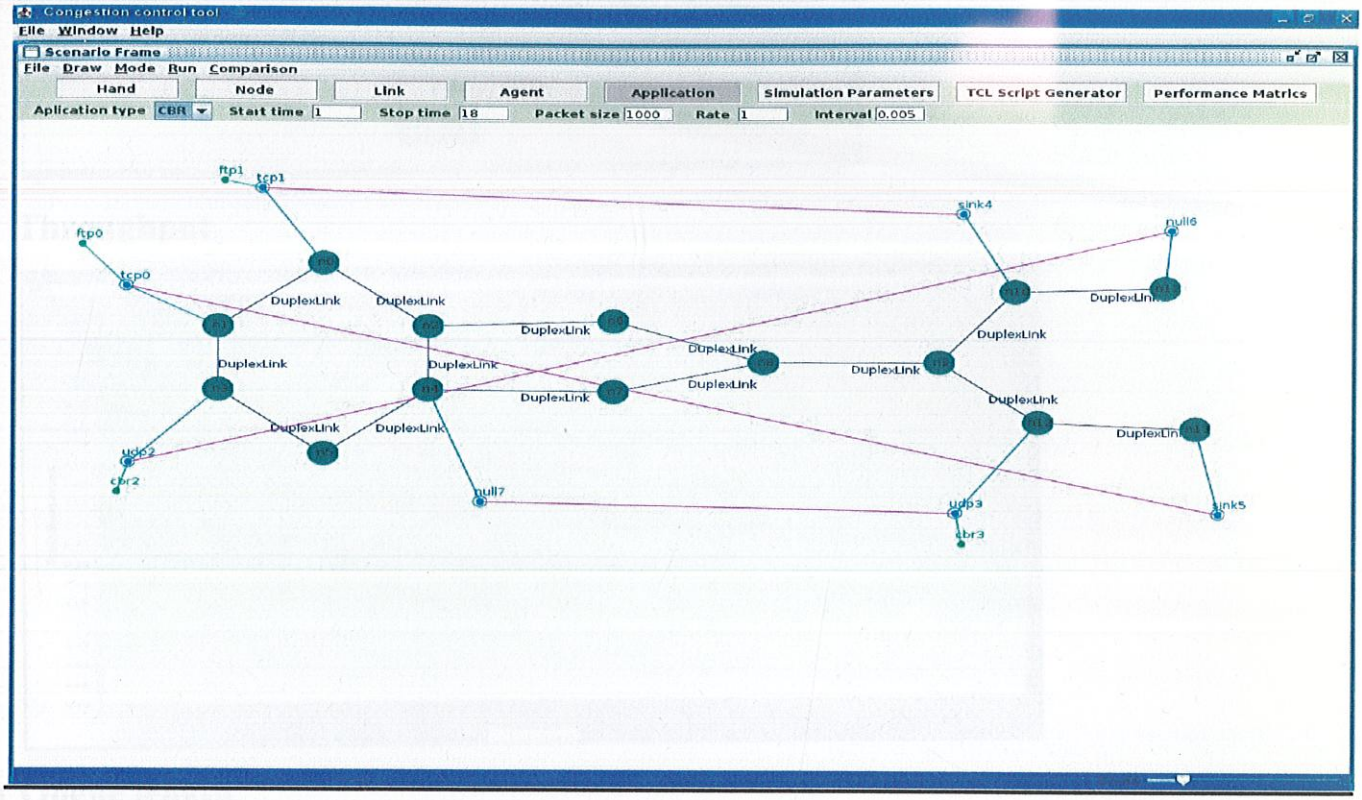
) **Queue Length**

) **Jitter**



8.2 Test Case 2

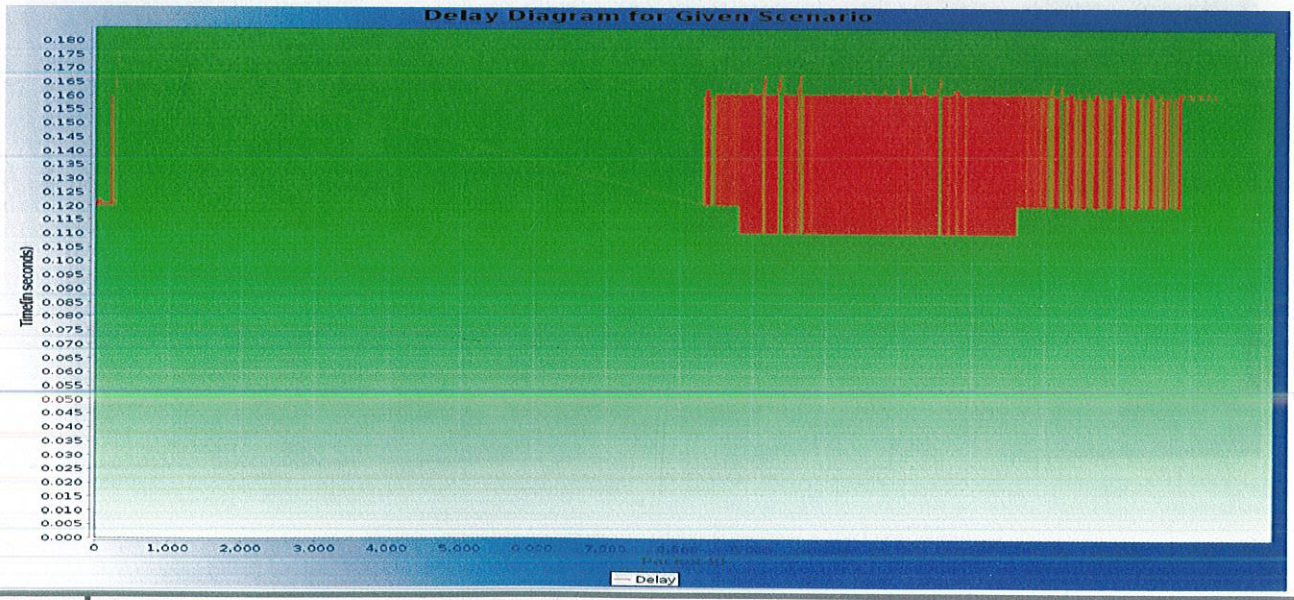
Simulation Scenario



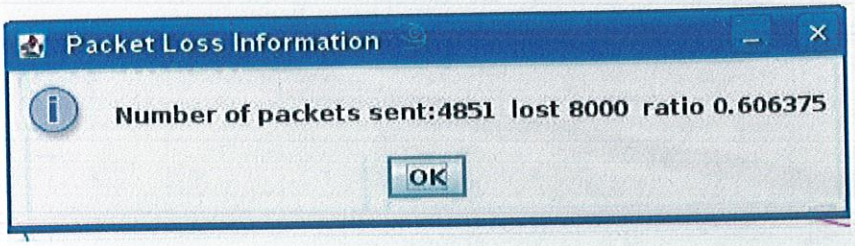
The tcl scipt and tr file is generated for blue, red and sfq.

Performance metric

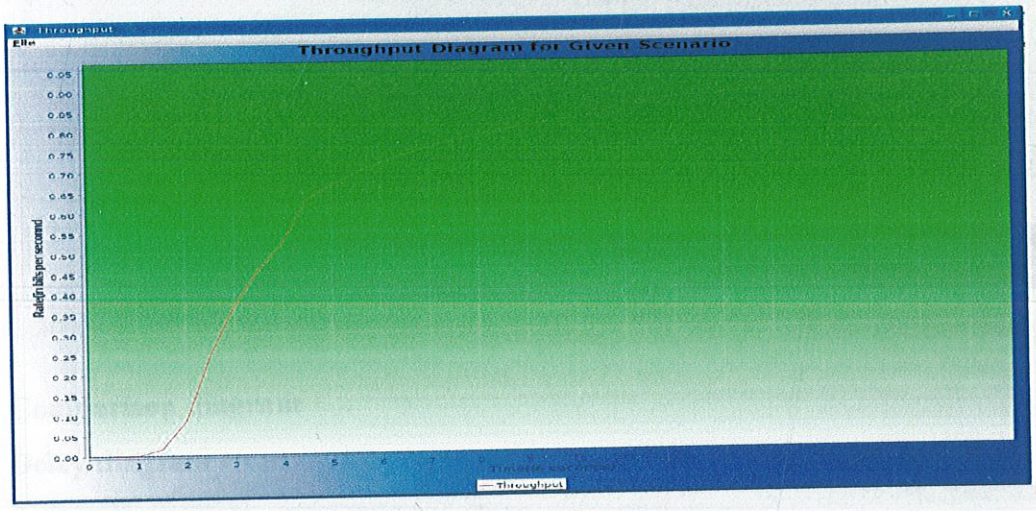
a) Delay



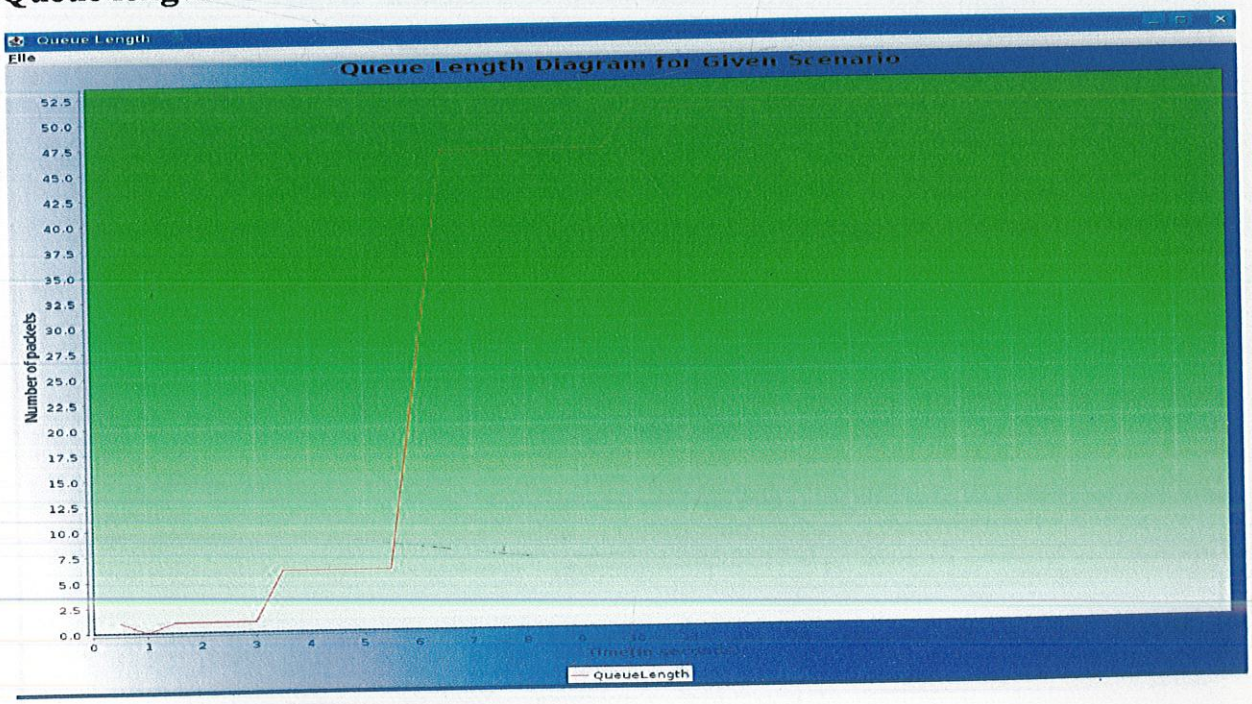
b) Packet Loss



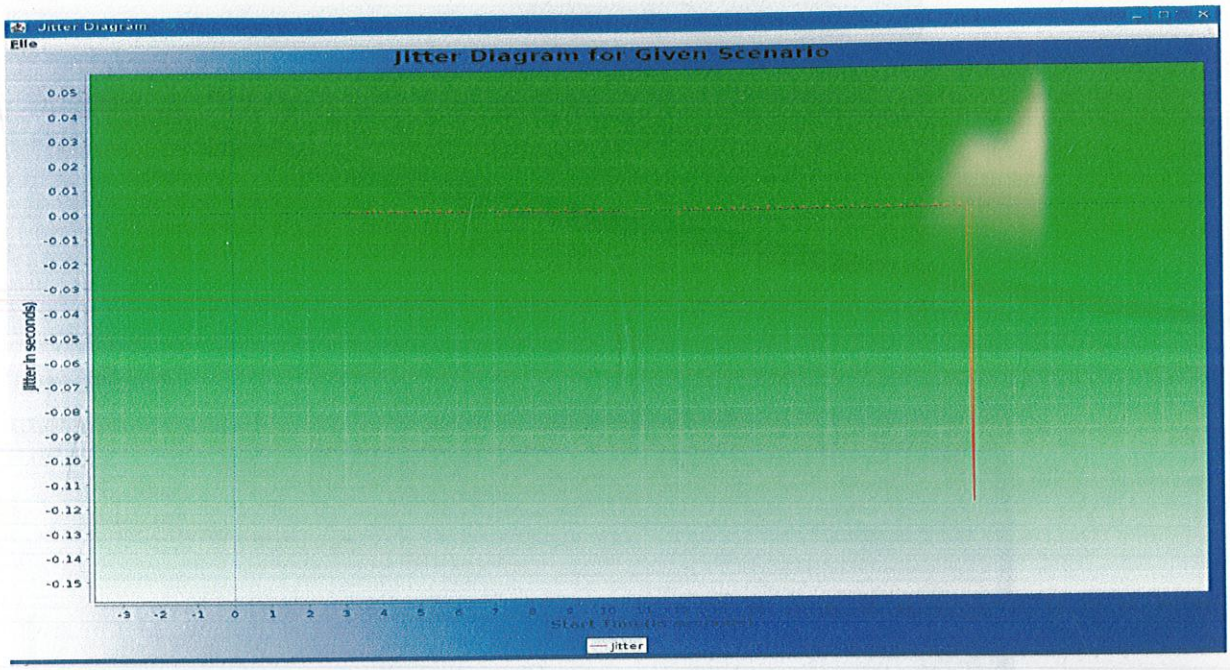
c) Throughput



d) Queue length

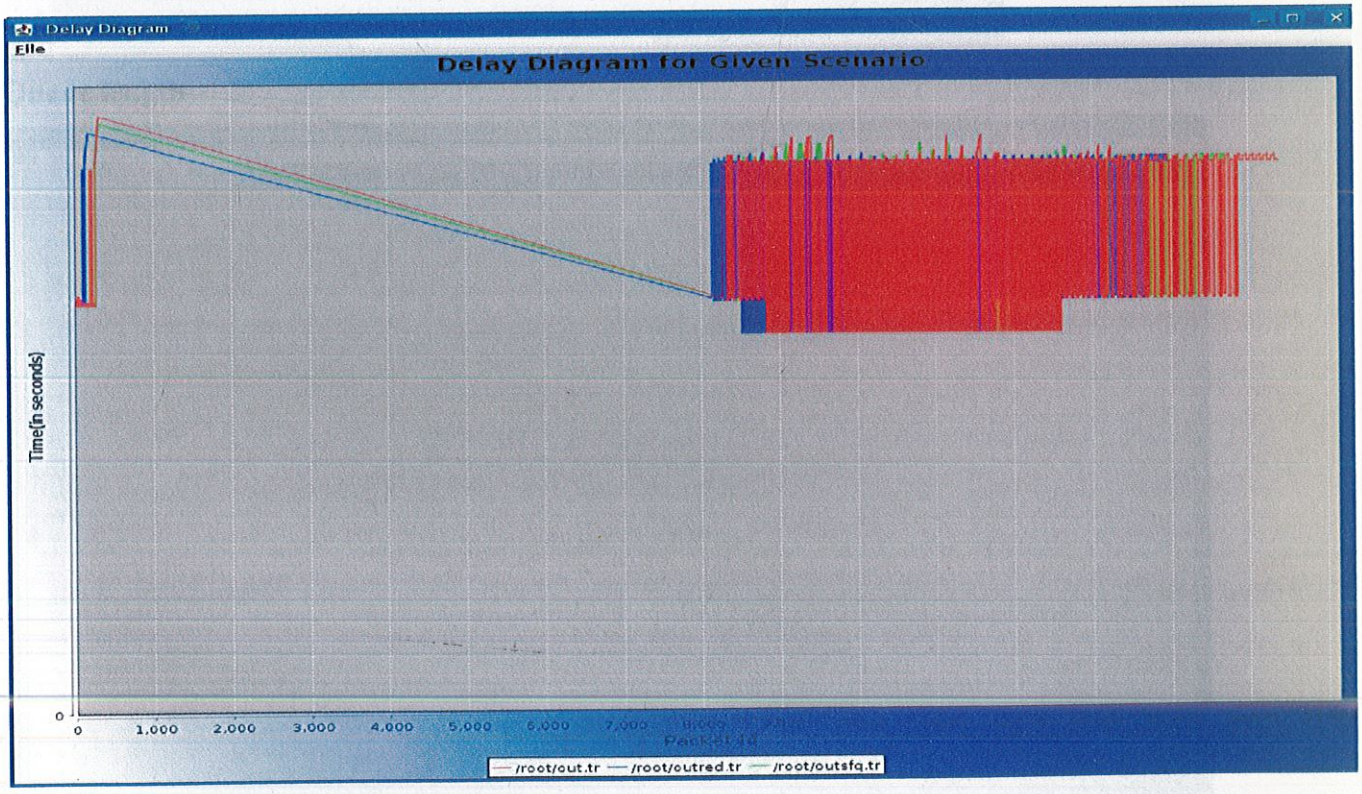


e) **Jitter**

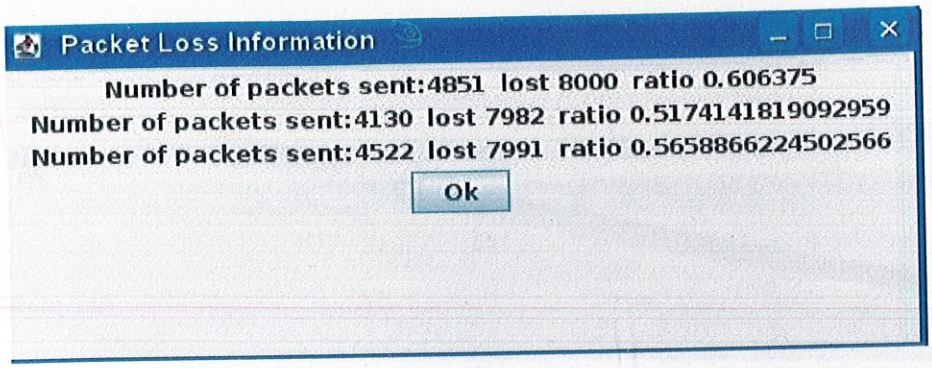


Comparison diagram

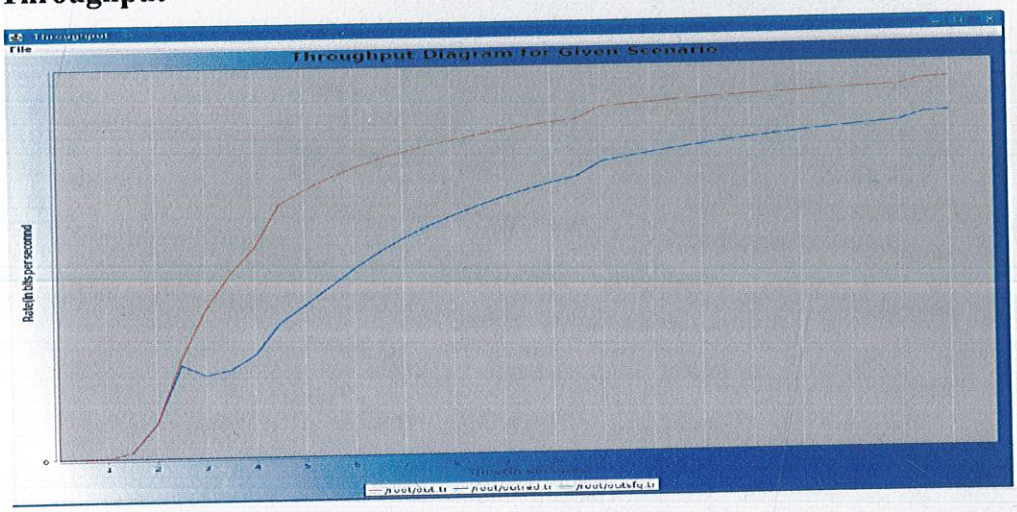
a) **Delay diagram**



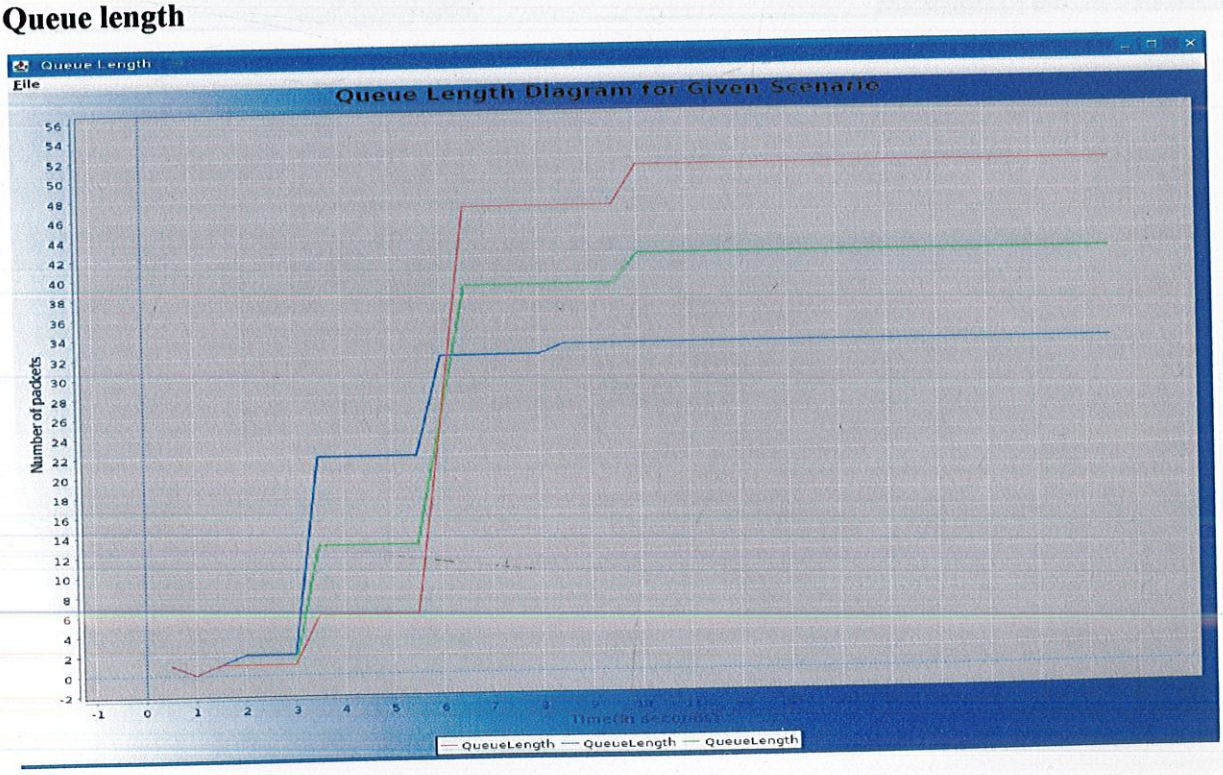
b) **Packet Loss**



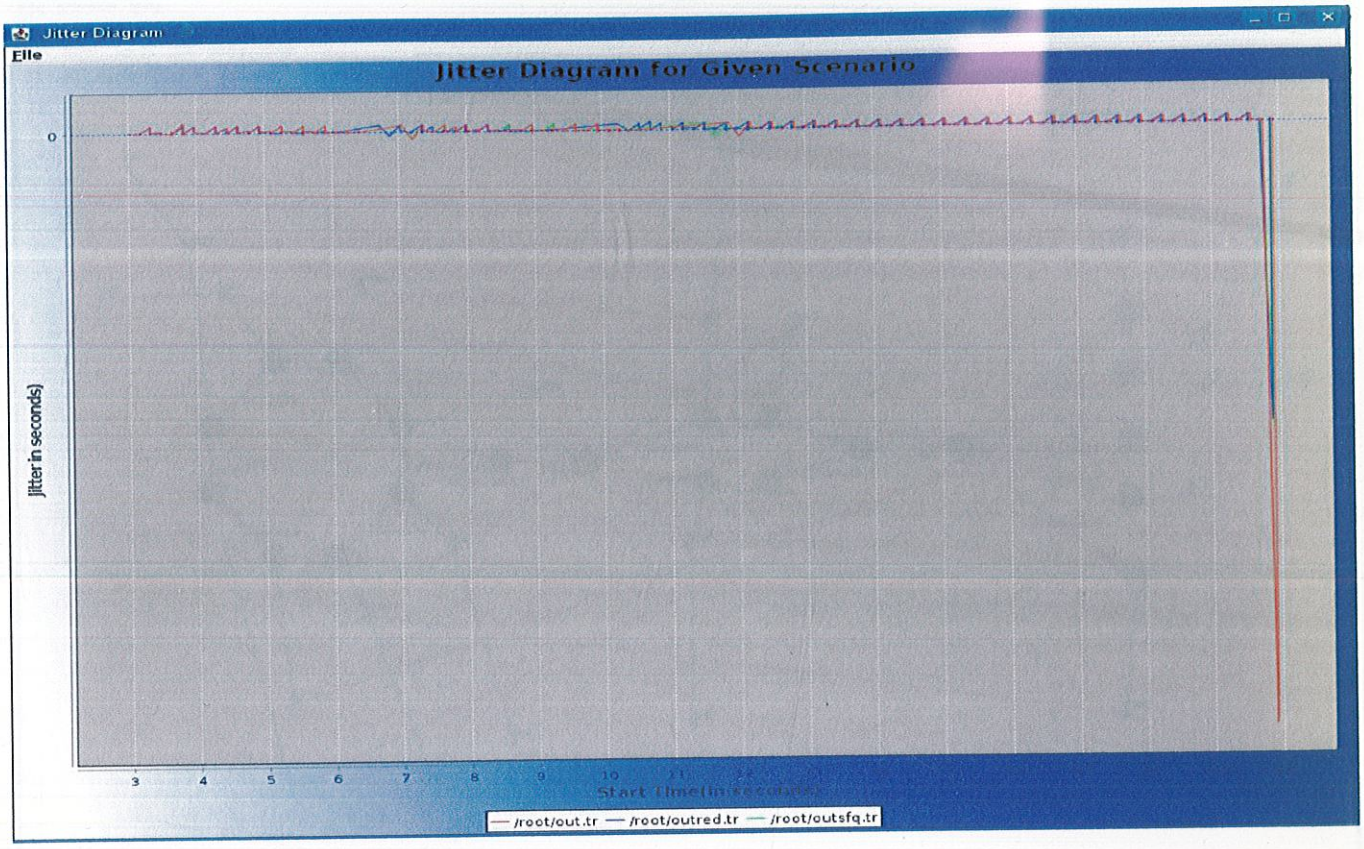
c) Throughput



d) Queue length

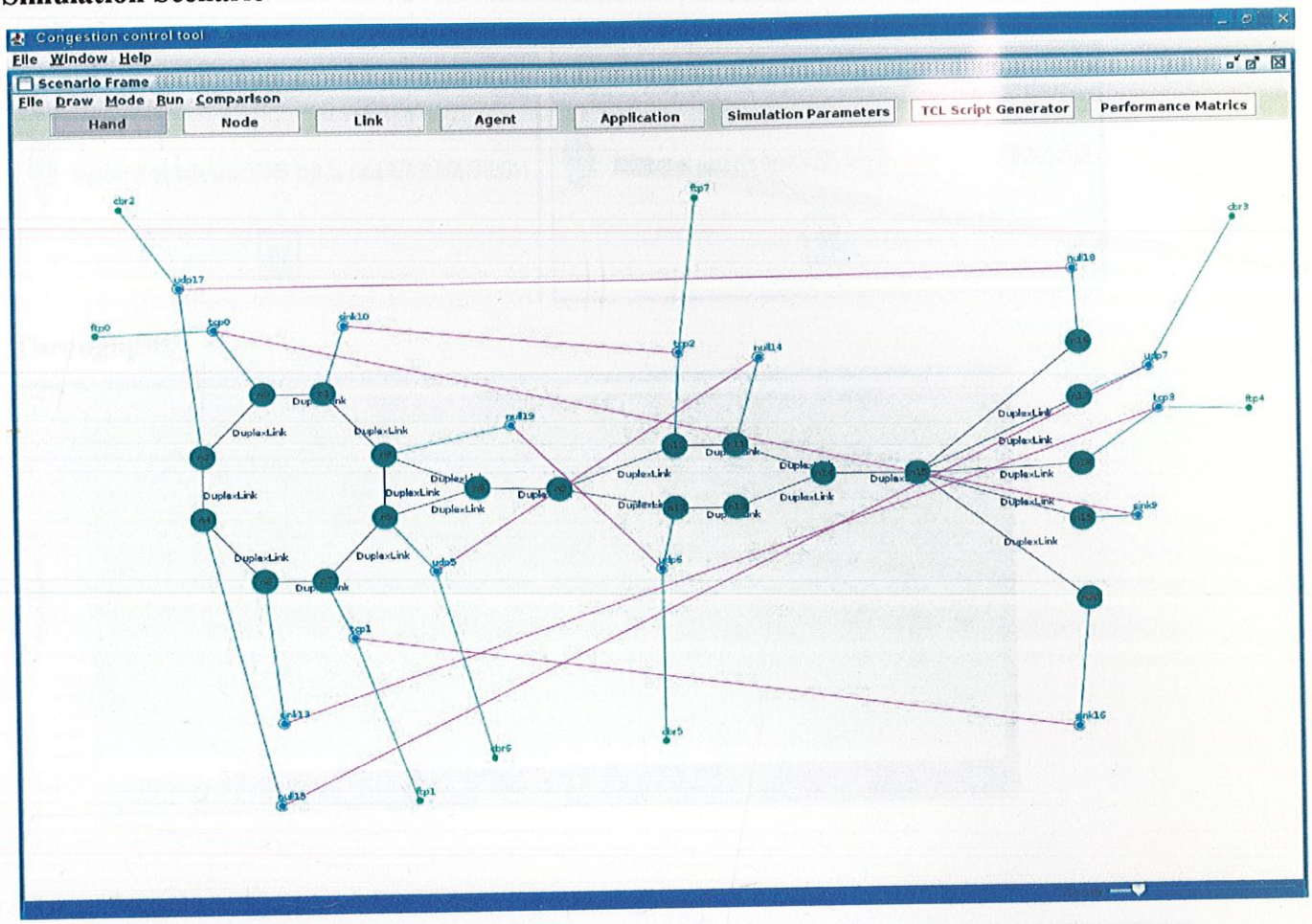


e) Jitter


Jitter Diagram
Elle
Jitter Diagram for Given Scenario
0
Jitter in seconds)
3
4
5
6
7
8
9
10
Start Time(in seconds)
/root/out.tr
/root/outred.tr
/root/outsfq.tr

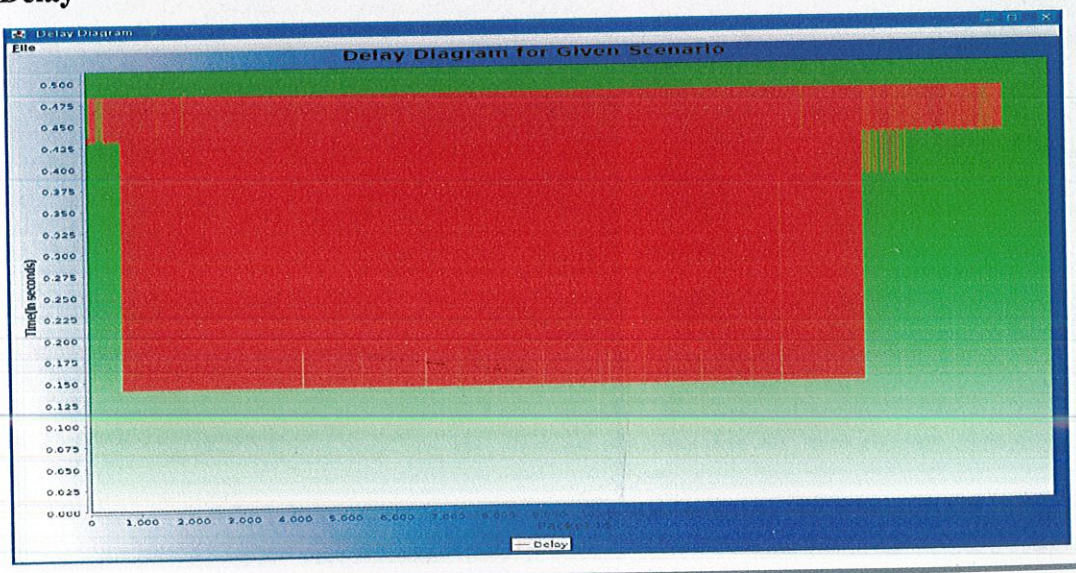
8.3 Test case3

Simulation Scenario

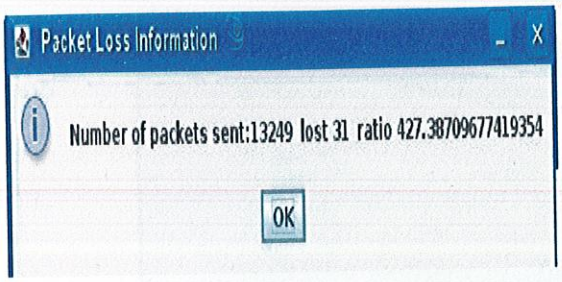


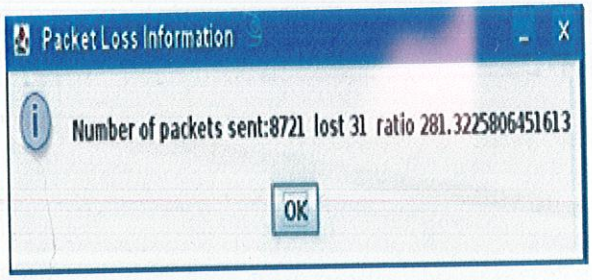
Performance metrics

a) **Delay**

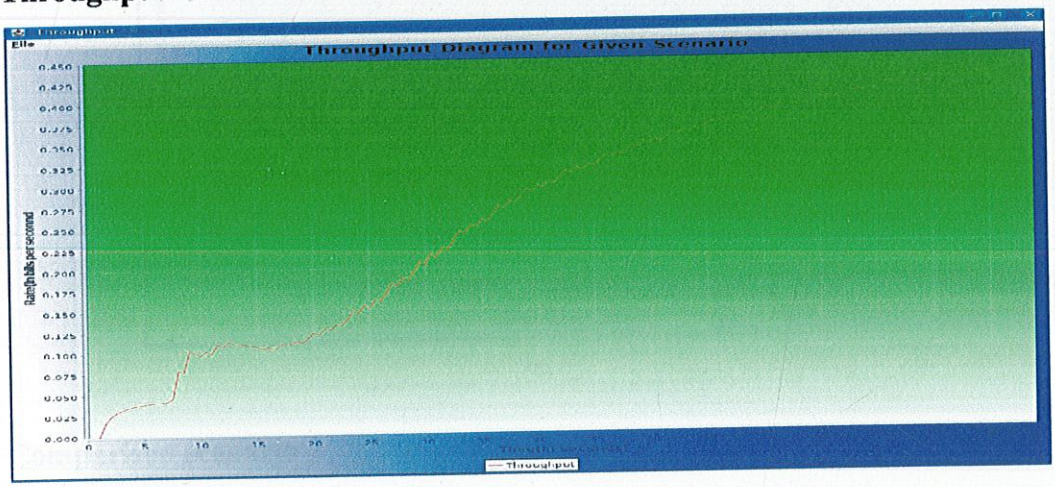


b) Packet loss

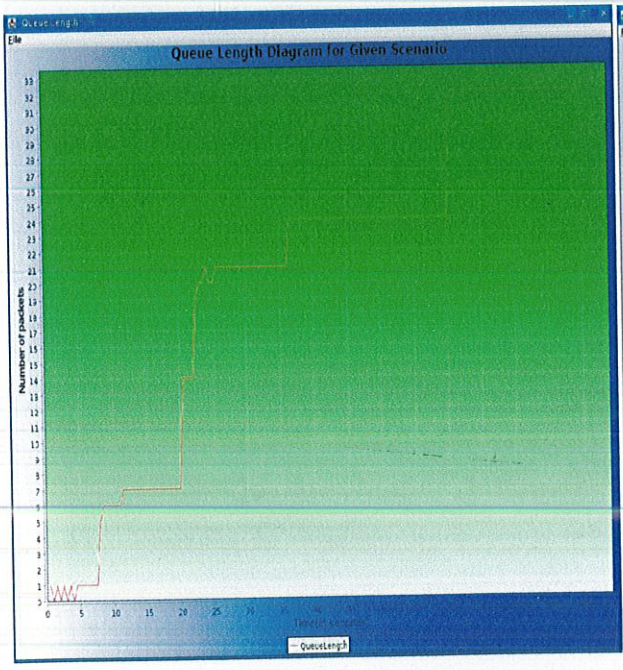


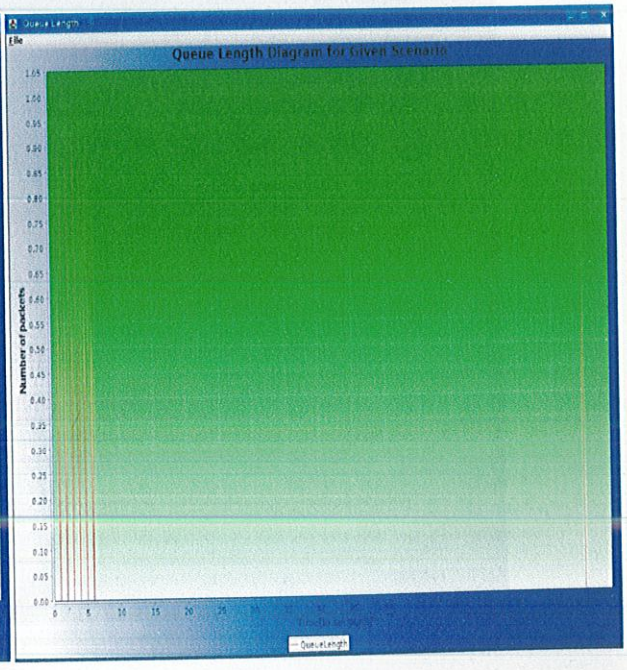


c) Throughput

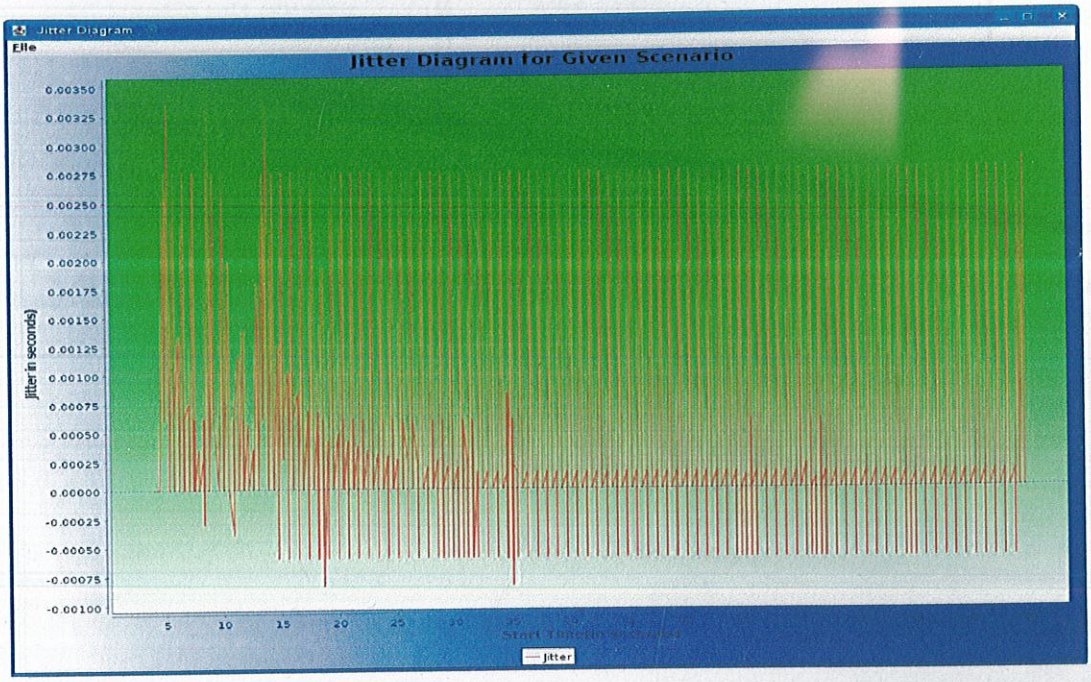


d) Queue length



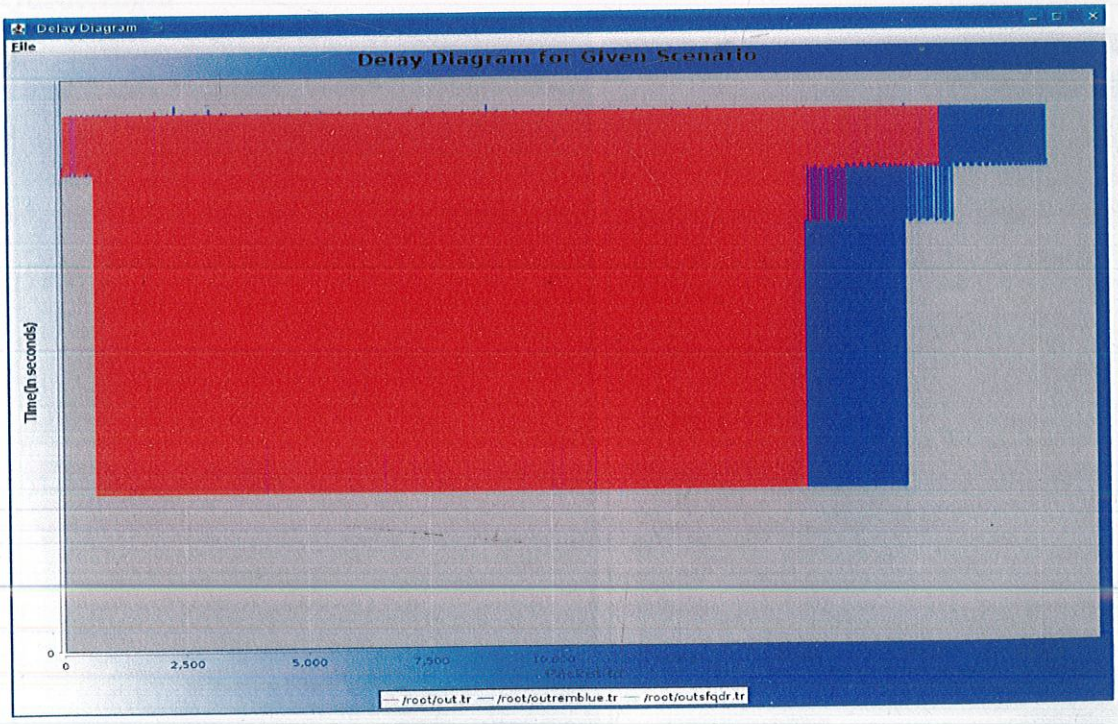


e) Jitter

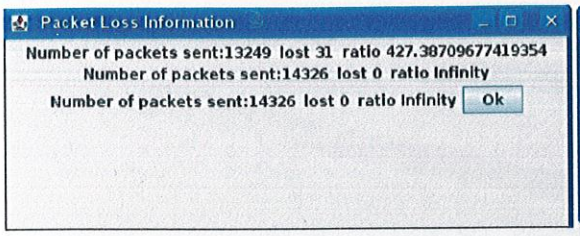


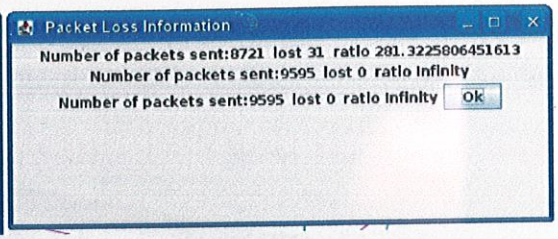
Comparison scenario

a) Delay

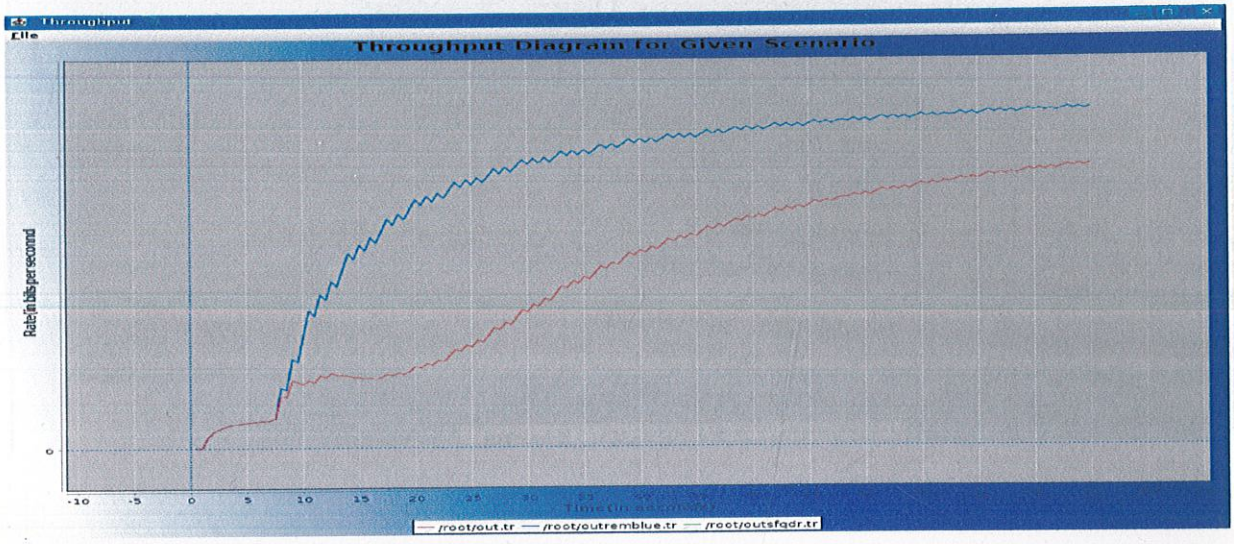


b) PacketLoss

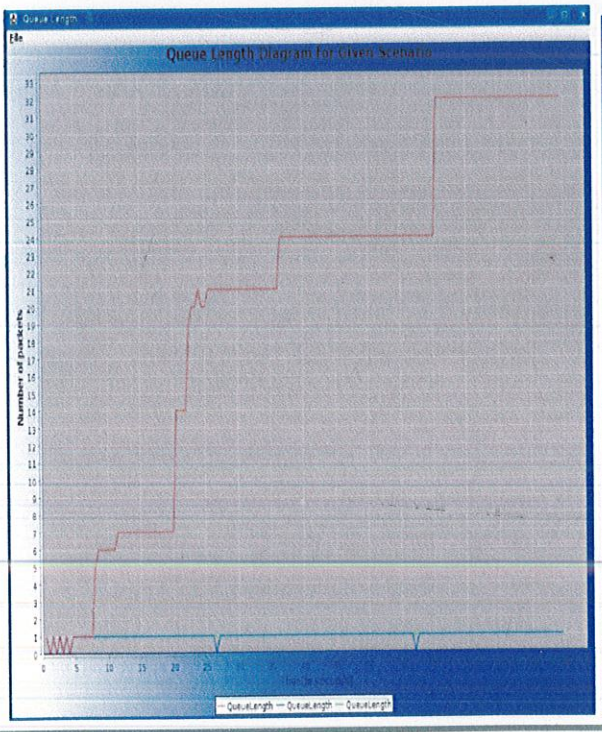


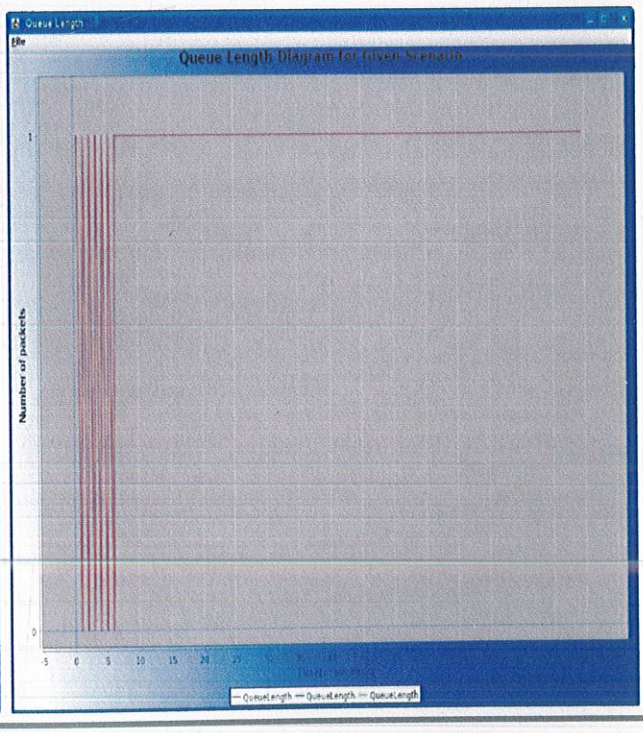


c) Throughput

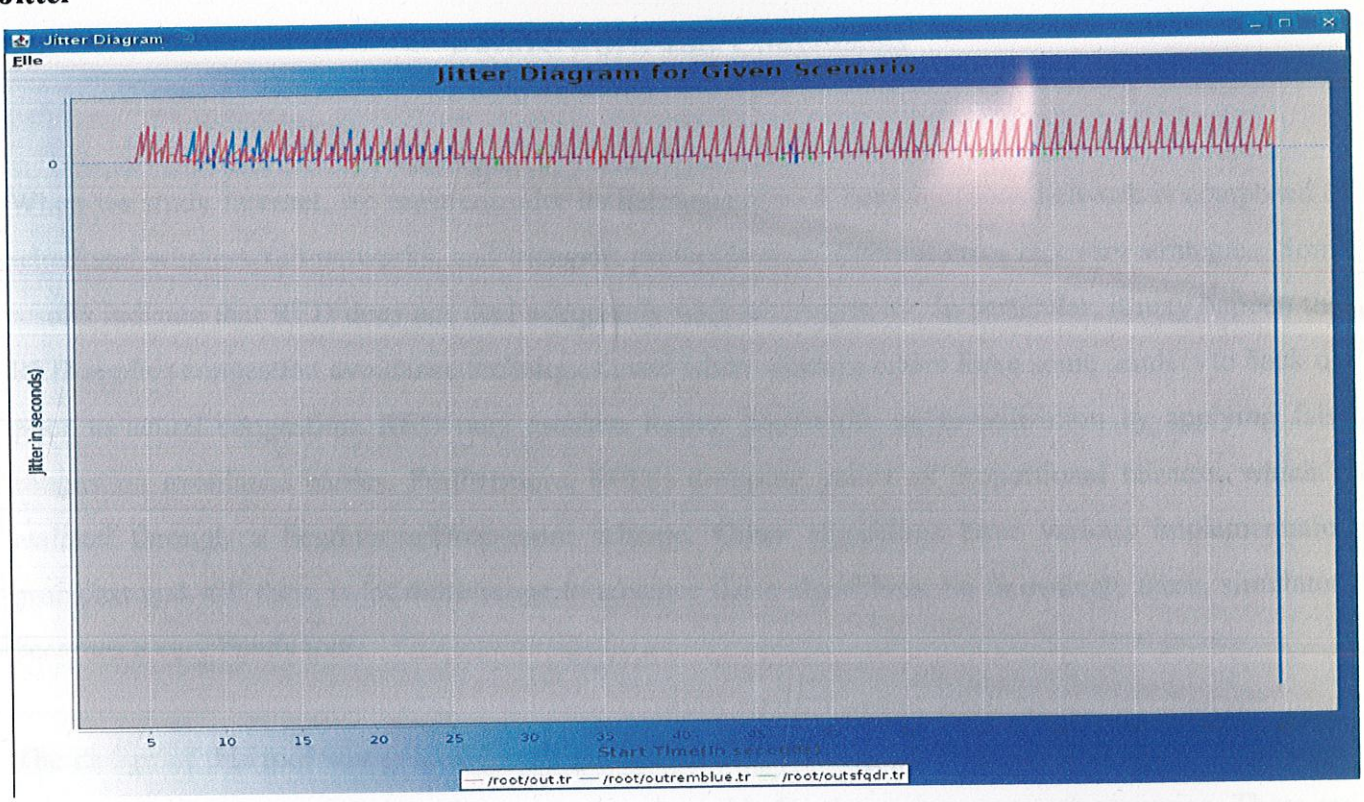


d) Queuelength





e) Jitter


Jitter Diagram
File
Jitter Diagram for Given Scenario
0
Jitter in seconds)
5
10
15
20
25
30
/root/out.tr
/root/outremblue.tr
/root/outsfqdr.tr

Future work and Conclusion

When we study Internet, we must consider its heterogeneity. A heterogeneous network is composed of wired and wireless sub-networks, and transport protocols with different error recovery strategies. Some results indicate that RED does not deal adequately with heterogeneity. In particular, it may happen that RED applies congestion avoidance techniques even when wireless errors force some senders to back off prior to actual congestion. RED may escalate further bandwidth under-utilization by applying false congestion avoidance tactics. Furthermore, RED's dropping policy of proportional fairness, which is realized through a Send-more/Drop-more scheme. Other algorithms have various implementation problems and still there is lot more scope to advance these algorithms. So to evaluate these, simulators becomes a very handy tool.

The design of this tool was guided by the goal of making its usage as easy as possible to compare the various congestion control algorithms as easy as possible for the various network scenarios. There are some other GUIs for network simulation available but our tools specially used for congestion control. So other settings are to be changed manually. As there are various limitation in it, so this tool can be integrated with a set of important tools which could comprise of complete and intuitive simulating environment. We can implement various other QoS architecture and mechanisms. We can also improve the scheduling of events for each object and to execute a usability validation of the tool by its users

In advanced to this software we can apply it to the wireless protocol to evaluate their performance. And the protocol here implemented can be directly appended. The simulation can be shown in the software itself.

Appendix A

Installing NS2 in LINUX platform

1. Download the tar file of ns allinone 2.xx , xx is the latest version
2. Untar the tar file where you want to deploy
3. In the terminal window open the nsallinone folder.
4. Check whether there is an executable file install is there
5. Type ./install
6. After installing, it will give you to set the environmental variable for the software, which are:

a) TCL_Library

b) LD_Libarary_PATH

c) PATH

7. To change the path the following step are given:

a) TCL_Library: export TCL_Library =$ TCL_Library: <PATH generated>

b) LD_Libarary_PATH: exportLD_Libarary_PATH =$ LD_Libarary_PATH: <PATH generated>

c) PATH: export PATH=$PATH: <PATH generated>

8. Path are given at the end of installing in the terminal window

9. You copy and paste in the above command

10. Put these command in the "profile.local" in the etc folder under root

Appendix B

Appendix B

Appending the new AQm algorithm in NS2

1. Add the .cc and .h file of the algorithm under the queue folder shown in ns hireacrchy
2. Modify the makefile and add the following

   Queue/ algo.o
3. Insert the default setting in "ns-default.tcl" file.
4. Now type makefile in terminal windoe. It will recompile the whole software.

BIBLIOGRAPHY

1. Floyd, S & Jacobson, V. *"Random Early Detection Gateways for Congestion Avoidance"*. IEEE/ACM Transactions on Networking August 1993

2. Jacobson ,V. "*Congestion avoidance and control*" .in 314-329, August 1988.Proceedings of SIGCOMM '88, pp.

3. Thomas Bonald, Martin May, Jean-Chrysostome Bolot. *"Analytic Evaluation of RED Performance"*. in proceedings of IEEE Infocom, March 2000

4. Youquan Zheng, Zhenming Feng. "*A New Active Queue Management Algorithm Based on the Rate and Burstiness Estimation*". Proceedings of IEEE ICC2002. Apr 2002, pp.2421-2425.

5. Patrick Brown, Denis Collange, Eitan Altman, Chadi Barakat, Emmanuel Laborde. "*Fairness Analysis of TCP/IP*". Proceedings of the 39" IEEEConference on Decision and Control

6. SHU-GANG LIU ."*Simulation and Evaluation of random early detection in congestion control*". Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming, 12-15 July 2008

7. Teunis J. Ott, T. V. Lakshman, and Larry H. Wong. "*SRED: Stabilized RED*". in Proceedings of IEEE INFOCOM, March 1999.9

8. Sanjeewa Athuraliya ,Victor H. Li, Steven H. Low, Qinghe Yin. "*REM: Active Queue Management*". *IEEE* Network, May/June 2001

9. Wu-chang Feng Kang G. Shin Dilip D. Kandlur Debanjan Saha. "*The BLUE Active Queue Management Algorithms". IEEE/ACM Transactions on Networking,* Vol. 10, No. 4, pp. 513-528, August 2002

10. Minseok Kwan and Sonia Fahmy. *"A Comparison of load based and queue based active Queue Management Algorithms"*. Dept. of Computer Science, Purdue University.

11. Ningning Hu, Liu Ren, Jichuan Chang. "*Evaluation of Queue Management Algorithms*" http://www.cs.cmu.edu/afs/cs/user/hnn/www/cs744_project/744-report.doc

12. Code for algorithm from University of California from http://www.cs.cmu.edu/

13. The ns Manual. The VINT Project. http://www.isi.edu/nsnam/ns/tutorial/index.html

14. S. Floyd. "RED web Page". http://www.aciri.org/floyd/red.html ,August 2001

15. P. Wu, "NS2 Scenarios Generator 2", available from http://sites.google.com/site/pengjungwu/nsg, 2008

16. E. Al-Shaer, "ns Bench –Graphical User Interface for Network Simulator", http://www.mnlab.cs depaul.edu/projects/nsbench/, 2005

17. http://nile.wpi.edu/NS/

18. Thomas Williams & Colin Kelley. *gnuplot An Interactive Plotting Program.* Manual 2007

19. James F. Kurose and Keith W. Ross. *Computer Networking :A top Down Approach featuring Internet.* ADDISON-WESLEY, Sept 1999

20. Andrew S. Tanenbaum. *Computer Networks, Fourth Edition.* Prentice Hall , March 17, 2003

21. Altman and Jimene. "*NS for Beginners*". Lecture Notes 2001