**IJCTA,** 9(10), 2016, pp. 4899-4911


# Network Traffic Management Using OMNeT++

***S. Kranthi *K. Pranathi *Dr. A.Srisaila**

***Abstract :*** The persistence of this paper is to talk about issues identified with Network Traffic Management. A moderately new class of system administration is quick turning into a need in business Networks. Moderate sized and extensive associations are discovering they should control network traffic behavior to guarantee that their key applications dependably get the assets they have to perform ideally. Controlling network traffic requires restricting data transfer capacity to specific applications, ensuring least transmission capacity to others, and checking movement with high or low needs. This activity is called as Network Traffic Management. This paper deals with the intent of minimizing delay over traffic guaranteed minimum bandwidth for all the available nodes in the network. The implementation details include the measurement of network traffic and monitoring statistics for analyzing the performance of the network by implementing the management techniques. A scenario is considered as an environment with '*n*' no. of wireless nodes forming a network with their own behavior, measuring the details and implementing necessary techniques with the advanced tools for the management of parameters optimally.

***Keywords :*** Availability, Network traffic, Packet, Routers, Sniffers.

## 1. INTRODUCTION

PC system is an information interchanges framework which interconnects PC frameworks at different distinctive locales. A system might be made out of any blend of LANs, or WANs. System activity can be characterized in various ways. However, in the least complex way we can characterize it as the thickness of information present in any Network. In any PC Network, there are a ton of specialized gadgets attempting to get to assets and in the meantime motivating solicitations to complete some work for some other gadget [1]. An arrangement of interconnected PCs and electronic peripherals, for example, printers is called PC system. This interconnection among PCs encourages data sharing among them. PCs may interface with each other by either wired or remote media. Parts of a system can be associated with each other distinctively in some style. By connectedness we mean either coherently, physically, or both ways. By and large, systems are recognized in light of their land range.

### A. Network Traffic

At a few times certain sorts of specialized gadgets might be occupied to react to the solicitation being made to them. So there is part of data trade in the Network in type of solicitation, reaction and control information. This information is fundamentally as a colossal number of parcels drifting around in the Network. This colossal measure of information goes about as a heap on the Network, which results in backing off the operations of other specialized gadgets. Because of this there is a ton of deferral in correspondence exercises. This at last results in blockage of the Network. This is the portrayal of Network Traffic in its least complex structure. . As it were we can say that Network activity is the heap on the specialized gadgets and the framework. This activity on the system has now brought about moderate sized and extensive associations understanding that they should control system movement

* Assistant Professor V R Siddhartha Engineering College Department of Information Technology Vijayawada, Andhra Pradesh, India pranathi.pamarthi@gmail.com, kranthisri41@gmail.com, sr.saila@gmail.com

conduct to guarantee that their key applications dependably get the assets they have to perform ideally [2]. Controlling system activity requires restricting data transmission to specific applications, ensuring least transfer speed to others, and stamping movement with high or low needs. This activity is called movement administration

### B. Traffic Classification

Traffic Classification is a robotized process which orders PC system movement as indicated by different parameters (for instance, in light of port number or convention) into various movement classes[3]. Each subsequent activity class can be dealt with distinctively keeping in mind the end goal to separate the administration suggested for the client (information generator/shopper). Administrators regularly recognize three expansive sorts of s ystem activity: Sensitive, Best-Effort, and Undesired.

1. **Latency Sensitive traffic :** Sensitive traffic is activity that the administrator has a desire to it convey on time. This incorporates VoIP, internet gaming, video conferencing, and web perusing. Activity administration plans are normally customized in a manner that the nature of administration, of these chose uses is ensured, or if nothing else organized over different classes of movement. This can be refined by the nonattendance of molding for this movement class, or by organizing touchy activity above different classes. System idleness is an outflow of the amount of time it takes for a parcel of information to get starting with one assigned point then onto the next. Idleness is the deferral from information into a framework to sought result. In a perfect world idleness is as near zero as could be allowed.
2. **Non Real-time traffic :** Best effort traffic is all different sorts of non-negative movement. This is movement that the ISP regards isn't delicate to Quality of Service measurements (jitter, bundle misfortune, idleness). A run of the mill case would be distributed and email applications. Movement administration plans are by and large customized so best-exertion activity gets what is left after delicate activity. For these focuses where the connection does not have adequate transmission capacity non ongoing administrations are considered to have lower need than others.
3. **Bursty traffic :** This classification is by and large constrained to the conveyance of spam and activity made by worms, botnets, and different malignant assaults. In a few systems, this definition can incorporate such movement as non-neighborhood VoIP (for example,Skype) or video spilling administrations to secure the business sector for the 'in-house' administrations of the same sort. In these cases, movement order systems distinguish this activity, permitting the system administrator to either obstruct this activity altogether, or extremely hamper its operation. High transfer speed is devoured in this movement. Information has a tendency to be bursty and there can be conflicting activity levels.
4. **Interactive traffic :** Wireless communication is innately telecast in nature. A unicast transmission can be heard by the objective collector, as well as by each other station in the area of the transmitter. Without a doubt, these stations (called forwarders from this point forward) normally disentangle all transmissions they hear and after that drop transmissions for which they are not the planned beneficiaries. By intelligent activity we mean movement, for example, TCP/VOIP in which there is a criticism process included. It is subjected to rivalry for data transmission and could bring about poor reaction times too.

## 2. REASONS TO MEASURE NETWORK TRAFFIC

**The accompanying are the purposes behind which we have measure the system movement.**

1. Service observing - ensuring things continue working.
2. Network arranging - choosing when more limit is required.
3. Cost recuperation - session times and movement volumes can give charging information.
4. Research - an improved understanding of what's going on should allow us to upgrade framework execution

The accompanying are the explanations behind which we have measure the system activity.

### A. Drivers for measurement

**There are number of other drivers strongly deals with requirement of measurement**

1. Pricing
2. Service level agreements
3. New services
4. Applications

### B. Approaches for traffic measurement

**There are some approaches for traffic measurement as follows :**

1. **Active Measurement of Traffic :** As name demonstrates, in this estimation approach clients or suppliers are specifically identified with the exercises to the estimation. There are number of various approaches to do this estimation like
   (*i*) Injection of tests into system by clients and suppliers
   (*ii*) Ping and Trace out the Path network and Roundtrip delay
   (*iii*) User-application execution as seen from hosts like Loss, Delay and Throughput
   (*iv*) Distribute on estimation servers makes the Probes are spread crosswise over lattice of ways through system to check adaptability and development of test movement.

   While displaying another application or organization to a framework it is critical to test the execution of the application before making it available for the customers. Dynamic measuring can be used to duplicate endless thusly it can help in finding for occasion what number of simultaneous customers a web server can advantage. For framework executives it is basic to know how well their framework performs with the objective that they perceive what sorts of organizations they can offer to their customers. Despite measuring execution, framework chairmen use dynamic/inert estimations to examine their framework. Sometimes there might be an issue in the framework that causes movement to be controlled the wrong way.
2. **Passive Measurement of Traffic :** In this methodology client is by implication manage framework utilizing some equipment or programming devices. Fundamentally some recorded information is utilized to locate the present activity estimation . The as of now utilized systems for this sort of estimation are as per the following
   **(*i*) Packet screens :** This can be accomplished by recording parcel headers on connection. It requires novel point of interest of convention and design concentrates on
   **(*ii*) Router/Switch movement measurements :** Analyzing switch or switch, the shrewd gadgets introduced at system, can give system interior conduct. Utilizing these gadgets we can get data about Packet drops, Counts and Flow measurements
   **(*iii*) Server and switch logs :** These records or logs can perform well work in measuring. They give synopses of dial session, directing overhauls or web-server log.

Traffic Management is sent at the WAN edge of an undertaking site. This is the place the rapid LAN meets the lower-speed WAN access join. The LANWAN crossroads is additionally where both Internet and intranet activity enter and leave the venture [5]. So it is the perfect spot to "agreeable" activity and to moderate the effect of noncritical and even suspicious movement grabbed on the Internet. Constraining or hindering the system assets accessible to silly or undesirable activity supports the execution of big business asset arranging (ERP), client relationship administration (CRM), and other vital, business basic applications.

In addition to monitoring traffic at the system edge, there are unadulterated execution issues to consider. The WAN access system is typically slower than the LAN, for the most part for budgetary reasons. Additionally Businesses pay repeating month to month charges for WAN administrations, while LAN data transfer capacity is free (after the underlying hardware speculations have been made). With fast LAN movement backing off at the lower-speed access circuit, the LAN-WAN edge is the place blockage is well on the way to happen. Another essential component to consider here is that most applications have been created to keep running on LANs. The

director then pushes a catch to spread those arrangements to the different system portions where they ought to be authorized. The fundamental issue of inactive estimations is the measure of information that is produced. On the off chance that we accept a gigabit join with a use of 60% (on IP-layer) and a normal parcel size of 300 bytes, then the catch rate is around 250000 bundles for each second. The movement rate is 75 mebibytes (MiBs) every second and along these lines the storage room required for one hour follow is 270000 mebibytes (= 270 gibibytes).

### C. Measuring tools

There are many tools available for measurement of traffic. They are listed according their categories. The Local Systems which includes NETSTAT, TCPDUMP, ETHREAL and NTOP. The Remote (END) System which having MIB, IF-MIB, SNMP and MRTG . The Routers are also having NETFLOW (CISCO) and LFAP (ENTERASYS). Lastly the SNIFFERS having RMON, RMON2 and NETRAMET.

While sniffers have been widely used to monitor network traffic at the data link layer and above, most commercial wireless sniffers are costly and are not flexible open source solution. Many popular network management and measurement tools are widely used to obtain the analysis of network performance using the above specified tools.

## 3. PROPOSED METHOD

This paper manages the plan of minimizing postponement over activity ensured least transmission capacity for all the accessible hubs in the system. The usage points of interest incorporate the estimation of system movement and observing insights for breaking down the execution of the system by actualizing the administration procedures.

### A. Components of omnet++

1. simulation kernel library
2. NED topology description language
3. OMNeT++ IDE based on the Eclipse platform
4. GUI for simulation execution, links into simulation executable (Tkenv)
5. command-line user interface for simulation execution (Cmdenv)
6. utilities (makefile creation tool, etc.)
7. documentation, sample simulations, etc.

Traffic management refers to the set of traffic controls within the network that regulate traffic flows for the purpose of maintain the usability of the network during the conditions of traffic problems. Traffic management has multiple goals. First, it attempts to distinguish the different types of traffic and handle each type in the appropriate way. For example, real-time traffic is forwarded with minimum delay. Comments are useful for generated documentation. Every simulation model is an instance of a compound module type. These models are developed completely independent of OMNeT++, and follow their own release cycles.

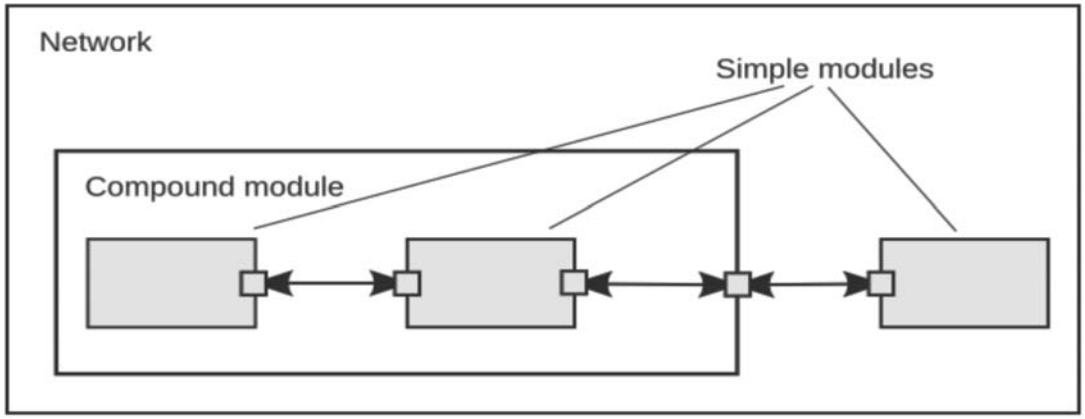


**Fig. 1. OMNeT++ architecture**

Modules communicate with messages that may contain arbitrary data, in addition to usual attributes such as a timestamp. Simple modules typically send messages via gates, but it is also possible to send them directly to their destination modules. Gates are the input and output interfaces of modules: messages are sent through output gates and arrive through input gates. An input gate and output gate can be linked by a connection. Connections are created within a single level of module hierarchy; within a compound module, corresponding gates of two submodules, or a gate of one submodule and a gate of the compound module can be connected. Connections spanning hierarchy levels are not permitted, as they would hinder model reuse.

### B. Building and running simulations

This segment furnishes bits of knowledge into working with OMNeT++ practically speaking. Issues, for example, model records and ordering and running reenactments are examined.

**An OMNeT++ model comprises of the accompanying parts :**

1. NED dialect topology depictions that portray the module structure with parameters, doors, and so forth. NED records can be composed utilizing any content manager, yet the OMNeT++ IDE gives superb backing to two-way graphical and content altering.
2. Message definitions (.msg records). You can characterize different message sorts and add information fields to them. OMNeT++ will make an interpretation of message definitions into undeniable C++ classes.
3. Simple module sources. They are C++ records, with .h/.cc postfix.

Reenactment projects are worked from the above segments. To start with, .msg documents are deciphered into C++ code utilizing the opp_msgc. Program. At that point all C++ sources are gathered and connected with the reenactment part and a client interface library to shape a reproduction executable or shared library. NED documents are stacked progressively in their unique content structures when the reproduction program begins. They may intuitively provoke the client for the quality, and they may likewise hold expressions referencing different parameters. Compound modules may pass parameters or articulations of parameters to their submodules. System copying, together with continuous recreation and equipment on top of it like usefulness, is accessible on the grounds that the occasion scheduler in the reenactment part is pluggable. The OMNeT++ appropriation contains a demo of ongoing reproduction and an oversimplified case of system imitating, enough to give you indicates in case you're into this region. Genuine system imitating with the INET Framework is in the line.

### C. NED Language

NED is a system portrayal dialect that is utilized to make the topology of systems in OMNeT++. The OMNeT++ IDE gives you a chance to make your topology either graphically or utilizing NED. NED gives the client a chance to announce basic modules, and interface and gather them into compound modules. The client can name some compound modules as systems; that is, independent reenactment models. Channels are another segment sort, whose occurrences can likewise be utilized as a part of compound modules.

### D. The ARP module

The ARP module executes the Address Resolution Protocol (RFC826). The ARP convention is intended to decipher a neighborhood convention location to an equipment address. Altough the ARP convention can be utilized with a few system convention and equipment tending to plans, by and by they are quite often IPv4 and 802.3 locations. The INET usage of the ARP convention (the ARP module) bolsters just IP address - >MAC address interpretation**.**

### E. Applications

The applications described in this section uses the services of the network layer only, they do not need transport layer protocols. They can be used with both IPv4 and IPv6.

**UDP Basic App**

The UDP Basic App sends UDP packets to a the IP addresses given in the destAddresses parameter. The application sends a message to one of the targets in each sendInterval interval. The interval between message and the message length can be given as a random variable. Before the packet is sent, it is emitted in the sentPk signal. The application simply prints the received UDP datagrams. The rcvdPk signal can be used to detect the received packets. The number of sent and received messages are saved as scalars at the end of the simulation.

**UDP Video Stream Svr**

This is the video stream server to be used with UDP Video Stream Cli. The server will wait for incoming "video streaming requests". When a request arrives, it draws a random video stream size using the videoSize parameter, and starts streaming to the client. During streaming, it will send UDP packets of size packetLen at every sendInterval, until videoSize is reached. The parameters packetLen and sendInterval can be set to 99 constant values to create CBR traffic, or to random values. Message exchange is not always the most suitable way of interaction between modules. For example, if you have a designated module for statistics collection (very much preferred over global variables!!!), then instead of sending the statistics updates to it via messages, it is more convenient to regard the statistics module as a C++ object and call a public member function of it created for this purpose.

## F. Statistics

**Statistics are collected on outgoing packets :**

1. **sent Pk :** packet object Statistics are collected on incoming packets:
2. **out Of Order Pk :** statistics of out of order packets. The packet is out of order, when has msgId and sourceId parameters and module received bigger msgId from same sourceID.
3. **drop Pk :** statistics of dropped packets. The packet is dropped when not out-of-order packet and delay time is larger than delay Limit parameter. The delay Limit = 0 is infinity.
4. **rcvd Pk :** statistics of not dropped, not out-of-order packets.
5. **end To End Delay:** end to end delay statistics of not dropped, not out-of-order packets.

## G. Algorithm

The algorithm that can be used to limit the bandwidth to the applications is the token bucket algorithm. The algorithm can be applied in the omnet++ environment and sufficient results are obtained according to the execution of the algorithm in the simulation environment for the described applications.

## TOKEN BUCKET ALGORITHM

The token bucket is a calculation utilized as a part of packet switched PC systems and information transfers systems. It can be utilized to watch that information transmissions, as bundles, fit in with characterized limits on data transfer capacity and burstiness (a measure of the unevenness or varieties in the movement stream).

**The token bucket calculation can be thoughtfully comprehended as takes after**

**A token is added to the pail consistently.**

1. The token can hold at the most b tokens. On the off chance that a token arrives when the container is full, it is tossed.
2. When a packet (system layer PDU) of n bytes arrives, n tokens are expelled from the pail, and the parcel is sent to the system.
3. If less than n tokens are accessible, no tokens are expelled from the can, and the bundle is thought to be non-conformant.

**Two sorts of markers are accessible :**

1. Single rate three shading marker
2. Dual rate three shading marker

**Two-Rate Three-Color Marking**

Two-rate three-shading policer arrangements incorporate a menial utmost—the top data rate (PIR)— that you set to the normal information rate for activity touching base at or withdrawing from the interface under top conditions.

Two-rate three-shading policer designs likewise incorporate a second burst measure—the crest burst size (PBS)— that characterizes the most extreme number of bytes for which the second token can collect unused top data transmission limit. Amid times of generally little crest activity (movement that touches base at or withdraws from the interface at normal rates that surpass the PIR), any unused top data transmission limit amasses in the second token can, yet just up to the most extreme number of bytes determined by the PBS. A movement stream is ordered yellow in the event that it surpasses the CIR and the accessible submitted data transmission limit amassed in the primary token pail however fits in with the PIR. Bundles in a yellow stream are verifiably set apart with medium-high PLP and afterward went through the interface. The token landing rate speaks to the single data transmission limit designed for the policer. You can determine the transfer speed limit as a flat out number of bits every second by including the data transmission limit bps proclamation. For a movement stream whose normal entry or takeoff rate surpasses the token landing rate, conformance to a two-shading policer rate limit relies on upon the tokens in the can. In the event that adequate tokens stay in the can, the stream is viewed as accommodating activity.

On the off chance that the basin does not contain adequate tokens, the stream is considered non-accommodating activity. Parcels in a non-adjusting movement stream (arranged as red activity) are taken care of as per policing activities. Contingent upon the setup of the two-shading policer, bundles may be verifiably disposed of; or the parcels may be re-set apart with a predefined sending class, a predetermined PLP, or both, and after that went through the interface. The token basin is at first filled to limit, thus the policer permits an underlying activity burst (consecutive movement at normal rates that surpass the token entry rate) up to the measure of the token pail profundity.

At mid times of moderately low movement (activity that touches base at or withdraws from the interface at normal rates beneath the token landing rate), unused tokens amass in the can, yet just up to the arranged token can profundity. The conduct of the meter is characterized as far as its mode and two token pails, C and E, with the rates, CIR and EIR, individually, and most extreme sizes CBS and EBS. The token cans C and E are at first (at time 0) full; *i.e.*, the token number Tc(0) = CBS and Te(0) = EBS. From there on, the token number Tc is augmented by one CIR times each second (up to CBS), and the token check Te is increased by one EIR times each second (up to EBS).

The genuine execution of a Meter doesn't should be demonstrated by above formal determination. The Marker mirrors the metering result by setting the DS field of the bundle to a specific code point. If there should arise an occurrence of the AF PHB the shading can be coded as the drop priority of the bundle. The trTCM is helpful in policing an administration, where a crest rate should be constrained independently from a conferred data rate. he Meter measures every bundle and passes the parcel and the metering result to the Marker. The Meter works in one of two modes: Color Blind mode and Color Aware mode In the Color Blind mode, the Meter predicts that the bundle stream is uncolored. In the Color Aware mode the Meter expect that the parcel stream has been pre shaded. The Marker (re)colors the parcel as indicated by the aftereffects of the Meter. Tokens are created at conferred data rate and are added to the token container. At the point when bundle arrives and there are sufficient tokens in the can, the parcel is considered to be in profile and the pertinent quantities of tokens are expelled from the can. On the off chance that there are insufficient tokens in the can the parcel is out of profile. The conditioner denote a parcel "green" when it is in profile and "red" when it is out of profile.

The conduct of the Meter is specifiedin terms of two token basins, P and C, and its modewith two rates PIR and CIR, separately. The greatest size of the token container P is PBS and the most extreme size of the token pail C is CBS. The token pails P and C are at first (at time 0) full, *i.e.*, the token check Tp(0) = PBS and the token tally Tc(0) = CBS. Later, the token number Tp is augmented by one PIR times each second up to PBS and the token tally Tc is increased by one CIR times each second up to CBS. The can contains tokens, each of which can speak to a unit of bytes or a solitary bundle of foreordained size.

## 4. RESULTS AND OBSERVATIONS

The system made out of 8 hosts and 6 switches. Hosts are associated with the switches by 100Mbps Ethernet joins. The switches are associated by PPP lines with lower datarates. Movement policing performed at the Ethernet interface of the edge switches (R0, R1, R4, R5). By changing the line module of the PPP interfaces, various bounce practices can be instrumented.

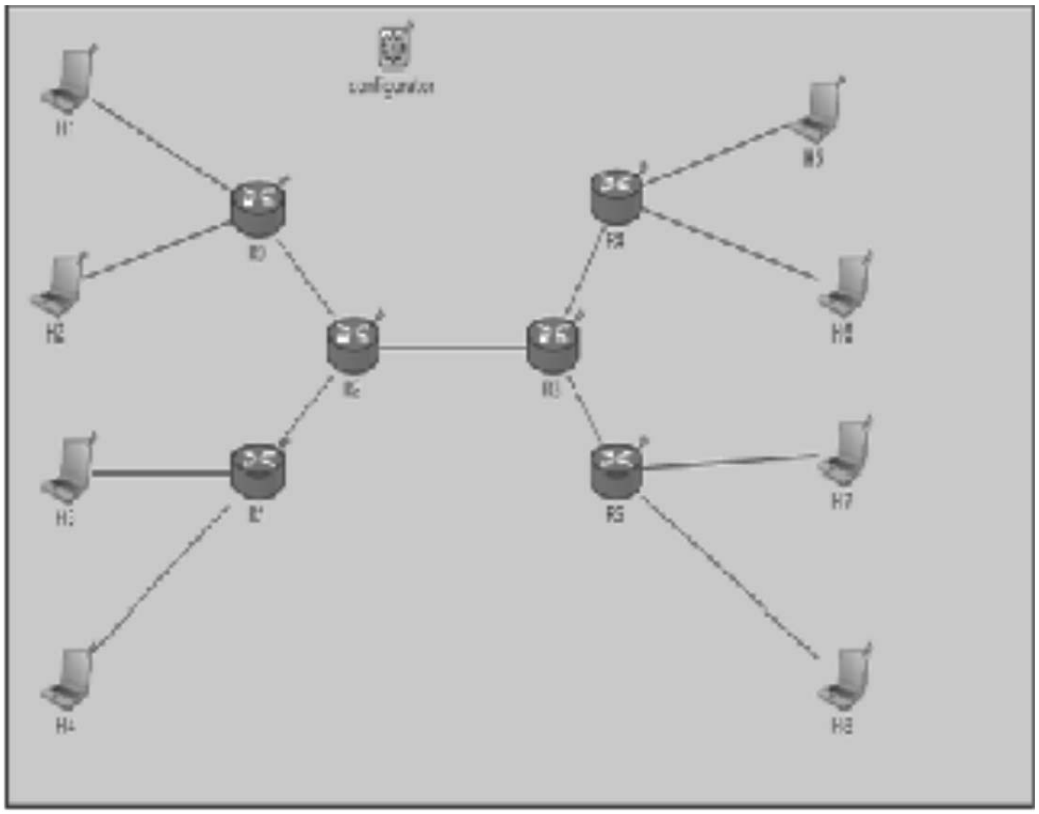

**Fig. 2. Network interface with hosts and routers**

Traffic flows from hosts H1-H4 to hosts H5-H8. The source hosts configured to send voice and video data to the destinations hosts.

### A. Router Behavior

Switches occasionally query Interval() send a General Query on each connected system for which this switch is a Querier. On startup the switch sends start up Query Count inquiries isolated by startup Query Interval. A General Query has unspecified Group Address field, a Max Response Time field set to queryResponseInterval, and is sent to the all-frameworks multicast address

### B. RED Dropper

The REDDropper module implements Random Early Detection ([FJ93]). It has n input and n output gates (specified by the numGates parameter). Packets that arrive at the $i^{th}$ input gate are forwarded to the $i^{th}$ output gate.

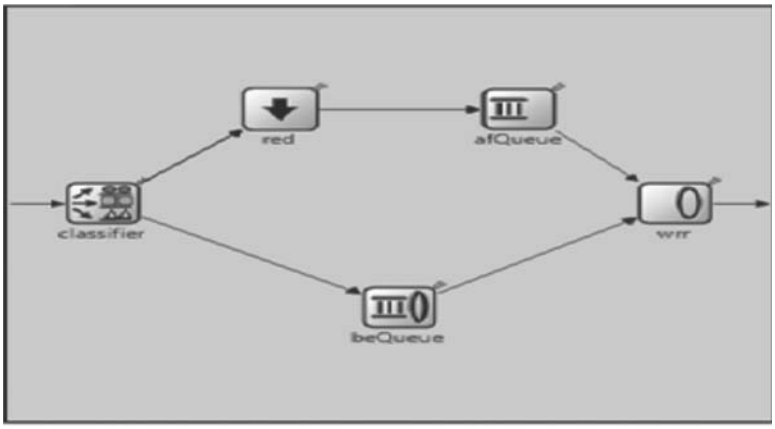


**Fig. 3. RED Dropper architecture**

### C. Priority Scheduler

The Priority Scheduler module executes a strict need scheduler. Bundles that touched base on in[0] has the most elevated need, then parcels landed on in[1], et cetera. On the off chance that more parcels accessible when one is asked for, then the one with most elevated need is picked. Bundles with lower need are transmitted just when there are no parcels on the inputs with higher needs. This is valuable for planning parcels in strict need request. Once in a while, two occasions may land at same time. In any case, the omnet++/omnest can deal with both the occasions in light of the landing time of every occasions.

### D. Weighted Round Robin Scheduler

The WRRScheduler module actualizes a weighted round-robin scheduler. The scheduler visits the info entryways thusly and chooses the quantity of bundles for transmission in view of their weight.

### E. Classifiers

Classifier modules have one data and numerous yield doors. They look at the got parcels, and forward them to the proper yield door in light of the substance of some segment of the bundle header.

### F. TokenBucketMeter

The TokenBucketMeter module executes a basic token can meter. The module has two yield, one for green parcels, and one for red bundles. At the point when a parcel arrives, the picked up tokens are added to the basin, and the quantity of tokens equivalent to the extent of the bundle are subtracted.

**Simple token bucket meter program :**

Token Bucket Meter ned file**:**

```
simple TokenBucketMeter
{
parameters:
@display("i = block/timer");
string interfaceTableModule;  // The path to the InterfaceTable module
string cir; // committed information rate
int cbs @unit(B);  // committed burst size
bool colorAwareMode = default(false); // enables color-aware mode
gates:
input in[];
output greenOut;
output redOut;
}
```

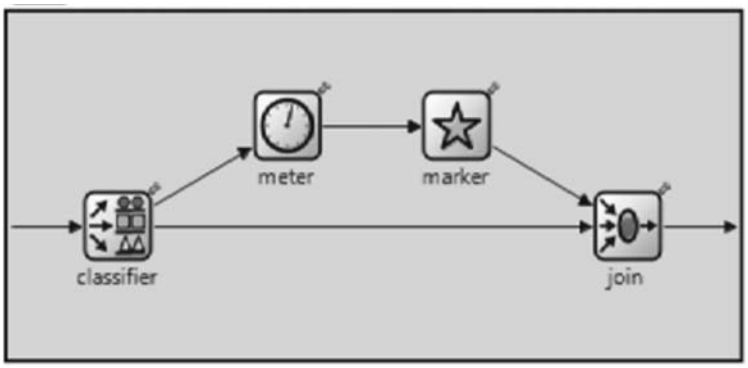


**Fig. 4. Traffic conditioner**

This experiment tests bandwidth utilization and flow isolation. Only hosts H1 and H2 generate traffic. H1 has an extra UDPBasicApp application, that generates a CBR traffic in addition to the voice and video streams. The rate of this extra traffic varies from 10kbps to 50kbps in 10kpbs steps in different measurements. Because the PPP links have only 300kbps datarate, the link between R2 and R3 is congested when the extra traffic is high enough.

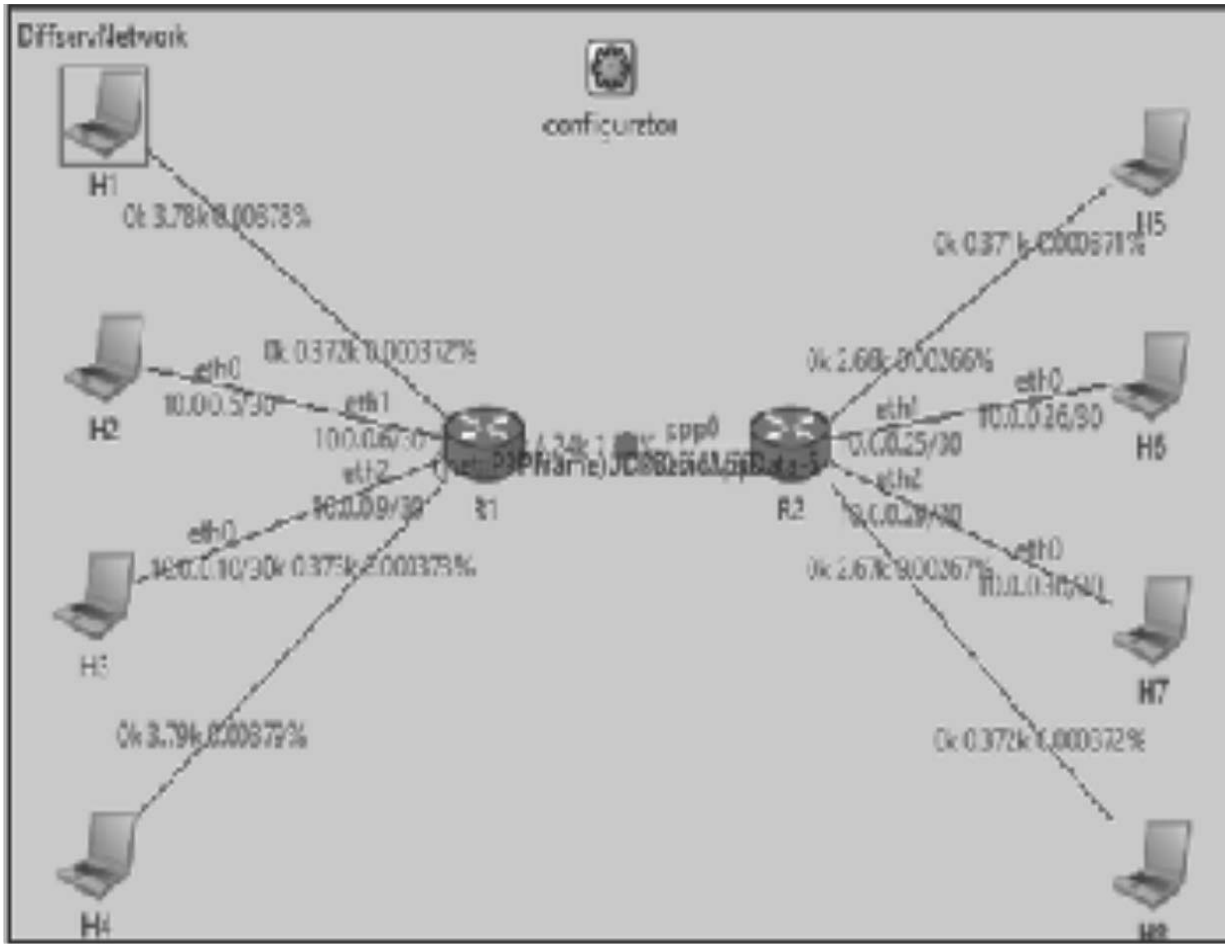


**Fig. 5. Simulating traffic between hosts**

## G. Result Recording:

OMNeT++ gives worked in backing to recording reenactment results, by means of yield vectors and yield scalars. Yield vectors are time arrangement information, recorded from basic modules or channels. You can utilize yield vectors to record end-to-end defers or round trek times of parcels, line lengths, queueing times, module state, join use, bundle drops, and so forth - anything that is helpful to get a full picture of what happened in the model amid the recreation run.

Yield scalars are rundown results, processed amid the recreation and worked out when the reenactment finishes. A scalar result might be a (whole number or genuine) number, or might be a factual outline involved a few fields, for example, check, mean, standard deviation, total, least, greatest, and so on., and alternatively histogram information.

**Results might be gathered and recorded in two ways :**

1. Based on the sign component, utilizing proclaimed insights;
2. Directly from C++ code, utilizing the recreation library

**Cmdenv Ini File Options**

Cmdenv-races to-execute indicates which reenactment runs ought to be executed. It acknowledges a comma-isolated rundown of run numbers or run number reaches, *e.g*. 1,3..4,7..9. In the event that the worth is missing, Cmdenv executes all runs that have ini document areas; if no runs are determined in the *ini* record, Cmdenv does one run. The - *r* summon line alternative overrides this *ini* record setting.

**Cmdenv can be executed in two modes, chose by the cmdenv-express-mode ini record choice:**

1. Normal (non-express) mode is for troubleshooting; itemized data will be composed to the standard yield (occasion flags, module yield, and so on).
2. Express mode can be utilized for long reenactment runs; just periodical announcements are shown about the advancement of the recreation

Measurements are pronounced in the NED documents with the @statistic property, and modules emanate values utilizing the sign instrument. The recreation structure records information by including uncommon result document essayist audience members to the signs. By having the capacity to pick what audience members to include, the client can control what to record in the outcome documents and what calculations to apply before recording. This additionally discloses how to instrument basic modules and channels for signs based result recording.

The signs approach takes into consideration estimation of total insights, (for example, the aggregate number of parcel drops in the system) and for actualizing a warm-up period without backing from module code. It additionally permits you to compose devoted insights gathering modules for the reproduction, likewise without touching existing modules.

The outcome recording-modes choice acknowledges one or more things as quality, isolated by comma. A thing might be an outcome recording mode (astonishment!), and two words with a unique importance, default what not:

1. A result recording mode implies any thing that may happen in the record key of the @statistic property; for instance, tally, whole, mean, vector((count-1)/2).
2. default remains for the arrangement of non-discretionary things from the @statistic property's record list, that is, those without question marks.
3. all implies all things from the @statistic property's record list, incorporating the ones with question marks.

The accompanying illustration just records the queueLength vectors and end To End Delay in voice App modules, and turns off the rest:

**. queue Length.vector-recording = genuine

**. voice App. end To End Delay.vector–recording = genuine

**. vector-recording = false

For the vector–recording–interims alternative, one can indicate one or more interims in the <startTime>..<stopTime> language structure, isolated by comma. <startTime> or <stopTime> should be given with estimation units, and both can be overlooked to signify the starting and the end of the recreation, separately.

| Event# | Time | Src/Dest | Name | Info |
|---|---|---|---|---|
| #62 | 1.544883177382 | H4 --> R1 | arpREQ | ARP req: 10.0.0.33=? |
| #75 | 1.544889037382 | R1 --> H4 | arpREPLY | ARP reply: 10.0.0.33= |
| #86 | 1.544894897382 | H4 --> R1 | UDPBasicAppData-0 | cPacket:500 bytes |
| #100 | 1.544939317382 | R1 --> R2 | UDPBasicAppData-0 | cPacket:500 bytes |
| #111 | 1.548813502304 | H1 --> R0 | arpREQ | ARP req: 10.0.0.2=? ( |
| #124 | 1.548819362304 | R0 --> H1 | arpREPLY | ARP reply: 10.0.0.2=0 |
| #135 | 1.548825222304 | H1 --> R0 | UDPBasicAppData-0 | cPacket:472 bytes |
| #149 | 1.548867402304 | R0 --> R2 | UDPBasicAppData-0 | cPacket:472 bytes |
| #158 | 1.561205984048 | R2 --> R3 | UDPBasicAppData-0 | cPacket:500 bytes |
| #167 | 1.575472650714 | R2 --> R3 | UDPBasicAppData-0 | cPacket:472 bytes |
| #174 | 1.577472650714 | R3 --> R5 | UDPBasicAppData-0 | cPacket:500 bytes |
| #182 | 1.584883177382 | H4 --> R1 | UDPBasicAppData-1 | cPacket:500 bytes |
| #195 | 1.584927597382 | R1 --> R2 | UDPBasicAppData-1 | cPacket:500 bytes |
| #204 | 1.590992650714 | R3 --> R4 | UDPBasicAppData-0 | cPacket:472 bytes |
| #215 | 1.592844616388 | H2 --> R0 | arpREQ | ARP req: 10.0.0.6=? ( |
| #228 | 1.592850476388 | R0 --> H2 | arpREPLY | ARP reply: 10.0.0.6=0 |
| #239 | 1.592856336388 | H2 --> R0 | UDPBasicAppData-0 | cPacket:172 bytes |
| #253 | 1.592874516388 | R0 --> R2 | UDPBasicAppData-0 | cPacket:172 bytes |
| #262 | 1.59373931738 | R5 --> H8 | arpREQ | ARP req: 10.0.0.50=? |
| #273 | 1.59374517738 | H8 --> R5 | arpREPLY | ARP reply: 10.0.0.50= |
| #286 | 1.59375103738 | R5 --> H8 | UDPBasicAppData-0 | cPacket:500 bytes |
| #303 | 1.600394516388 | R2 --> R3 | UDPBasicAppData-0 | cPacket:172 bytes |

**Fig. 6. Event log analysis**

When you are running several simulations with different parameter settings, you'll usually want to refer to selected input parameters in the result analysis as well — for example when drawing a throughput (or response time) versus load (or network background traffic) plot.

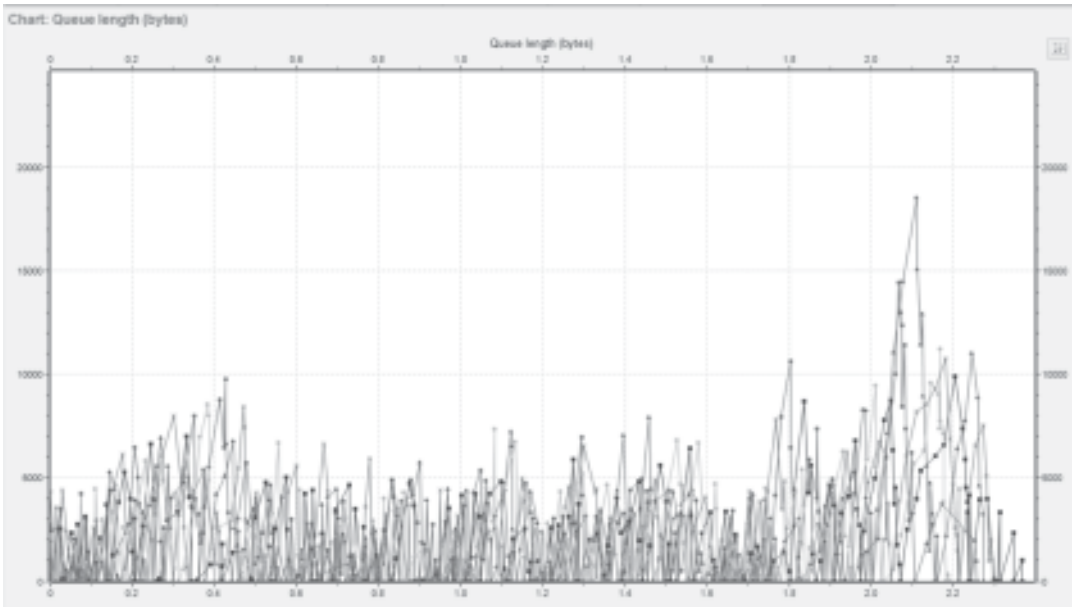


**Fig. 7. Queue length in bytes**

Average throughput or response time numbers are saved into the output scalar files, and it is useful for the input parameters to get saved into the same file as well. For convenience, OMNeT++ automatically saves the iteration variables into the output scalar file if they have numeric value, so they can be referred to during result analysis.

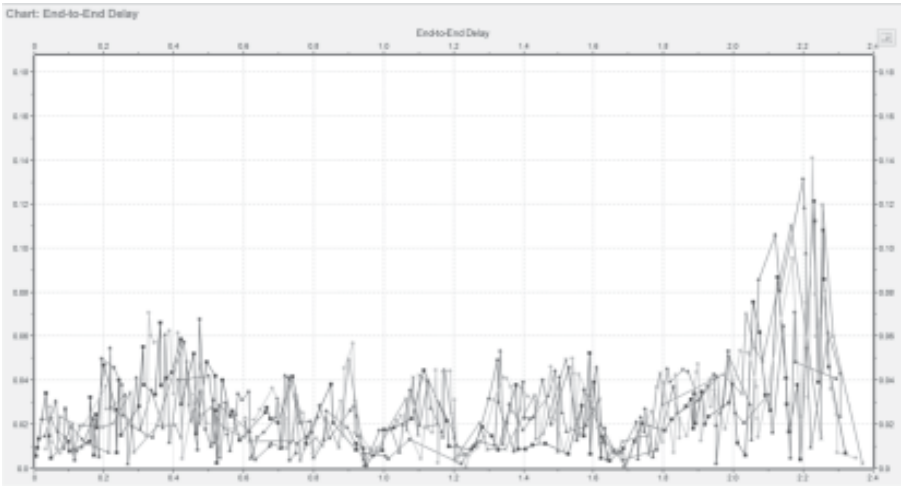


**Fig. 8. End to end delay**

By opening the inspector window for the transmission-time object one can view the plot of the recorded data. The plot only shows the recorded data since the opening of the window, so the simulation should be started again, but this time using the fast toolbar button.

## 5. CONCLUSION

In today's evolving situation, where the ordinary method for doing things no more holds great, associations are quick understanding that all together that they stay in venture with others in the race, they should grasp this idea of Network Management. Overseeing movement in this system fragment helps disseminated associations that rely on upon the WAN to serve remote clients with concentrated assets. Doing as such is a sensibly basic matter. Presently, neighborhood systems are by and large free from blockage and fall under the aggregate control of an interior IT office So it is the perfect spot to "agreeable" activity and to relieve the effect of noncritical and even suspicious movement grabbed on the Internet. Restricting or hindering the system assets accessible to unimportant or undesirable activity supports the execution. Likewise the way in which both the extent of systems and the information which rides on them is expanding by the day, it has ended up basic to screen the sort of movement streaming, needs it and after that deal with the activity as needs be.

## 6. REFERENCES

1. K. Dresner and P. Stone, "Multi-agent traffic management: An improved intersection control mechanism," in Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi Agent Systems, Utrecht, The Netherland, July 2005.
2. Roughan, M. "Simplifying the synthesis of Internet traffic matrices" Computer Communication Review 35 93–96, 2005.
3. M. Yuksel, B. Sikdar, K. S. Vastola and B. Szymanski, "Workload generation for ns Simulations of Wide Area Networks and the Internet", Proc. Of Communication Networks and Distributed Systems Modeling and Simulation Conference, pp 93-98, San Diego, CA, USA, 2000.
4. Roughan, M. and Kalmanek, C. R. "Pragmatic modeling of broadband access traffic". Computer Communications 26 (8) 804–816, 2003.
5. Tomás, Vicente R. et al. "New technologies to work with traffic management plans.",Traffic Technology International-Annual review, 2003.
6. NetVeda LLC, "Policy.Net: Policy Based Network Traffic Managementlice" http://www.netveda.com/ downloads/ PolicyNet_Whitepaper.pdf, 2003-2005
7. Internet Traffic Management, http://www.sics.se /cna/tm/
8. Wikipedia, http://en.wikipedia.org/wiki/ _management
9. BusinessWeek, http://whitepapers.businessweek.com /rlist/term/Network-Traffic-Management.html
10. http://arxiv.org/ftp/arxiv/papers/0912/0912.3972.pdf