Simulating Using *ns-2*

Ankit Verma*
Chandan Kumar**
Meenakshi Vyas***

1. Introduction: *ns-2* is called event simulator because *ns-2* simulator has list of events. Process includes beginning of event and run it until it is finished. After the completion of one event, another event starts. Each event happens in a short period of time [1]. *ns-2* is an object oriented simulator, written in C++, with an OTcl interpreter as a front-end. The simulator supports a class hierarchy in C++ and a similar class hierarchy within the OTcl interpreter. The two hierarchies are closely related to each other. There is a one-to-one correspondence between a class in the interpreted hierarchy and compiled hierarchy [2]. The root of this hierarchy is the class Tcl Object. *ns-2* uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulation of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets.

ns-2 meets both of these needs with two languages, C++ and OTcl. C++ is fast to run but slower to change, making it suitable for detailed protocol implementation. OTcl runs much slower but can be changed very quickly. *ns-2* is a event scheduler and all events are handled under c++ compiled hierarchy. The scheduler keeps ordered data structure and executes them one by one. Commands for simulation is written in tcl language (Tool command language), it is used because it is a language with a very simple syntax and allows very easy linking with other [2].

2. Simulation Using *ns-2:* Simulation using *ns-2* involves five important steps:

A. **Creating the event scheduler**:-*ns-2* is a discrete event network simulator. The scheduler runs by selecting the next earliest event, executing it until its completion and returns to execute the next event. Unit of time used by scheduler is seconds if more than one event is scheduled to execute at same time, then their execution is performed on first scheduled first dispatched manner [1].

B. **Creating network**:-Creating network requires two important steps i,e.
- **Creating nodes**:-one aspect of creating a topology in *ns-2*, *i.e.*, creating the nodes. The class provides instance procedures to create and manage the topology, and internally stores references to each element of the topology[1].
- **Creating link** :- This is the second aspect of defining the topology *ns-2* supports a variety of other media, including an emulation of a multi-access LAN using a mesh of simple links, and other true simulation of wireless and broadcast media. The CBQlink is derived from simple links and is considerably more complex form of link, as with the node being composed of classifiers, a simple link is built up from a sequence of connectors [1].

*Maharaja Ranjit Singh College of Professional Sciences, Indore, India

C. **Creating routes**:-There are basically two types of routing:

- **Unicast routing**:-The user level simulation script requires one command to specify the unicast routing strategy or protocols for the simulation[1]. A routing strategy is a general mechanism by which *ns-2* will compute routes for the simulation .There are four routing strategies in *ns-2*: Static, Session, Dynamic and Manual[5]. Conversely, a routing protocol is a realization of a specific algorithm. Currently, Static and Session routing use the Dijkstra's all-pairs SPF algorithm; one type of dynamic routing strategy is currently implemented: the Distributed Bellman-Ford algorithm in *ns-2*, we blur the distinction between strategy and protocol for static and session routing, considering them simply as protocols1. rtproto{} is the instance procedure in the class Simulator that specifies the unicast routing protocol to be used in the simulation [2].
- **Multicast routing**:-Multicast forwarding requires enhancements to the nodes and links in the topology. Therefore, the user must specify multicast requirements to the Simulator class before creating the topology when multicast extensions are thus enabled, nodes will be created with additional classifiers and replicators for multicast forwarding and links will contain elements to assign incoming interface labels to all packets entering a node[1]. A multicast routing strategy is the mechanism by which the multicast distribution tree is computed in the simulation. *ns-2* support three multicast route computation strategies: centralized, dense mode (DM) or shared tree mode (ST). The method mrtproto{} in the Class Simulator specifies either the route computation strategy, for centralized multicast routing, or the specific detailed multicast routing protocol that should be used[1].

D. **Creating traffic:-**After defining nodes and links between we should now we make a traffic flow through them for that we need to define routing, the agent and application that uses them. The two agents that we frequently use in simulation are:

- **UDP**:-A UDP agent accepts data in variable size chunks from an application, and segments the data if needed. UDP packets also contain a monotonically increasing sequence number and an RTP timestamp. Although real UDP packets do not contain sequence numbers or timestamps, this sequence number does not incur any simulated overhead, and can be useful for trace file analysis or for simulating UDP-based applications. The default maximum segment size (MSS) for UDP agents is 1000 byte[7].
- **TCP**:-There are two major types of TCP agents: one-way agents and a two-way agent. One-way agents are further subdivided into a set of TCP senders (which obey different congestion and error control techniques) and receivers ("sinks"). The two-way agent is symmetric in the sense that it represents both a sender and receiver[2].

3. Example of Simulation to obtain 2 node network.

```
set ns [new Simulator]
set tf [open out.tr w]  #creation of trace file
$ns trace-all $tf
set nf [open out.nam w]
```

```
$ns namtrace-all $nf
set qsize [open queueSize.tr w]
set n0 [$ns node] # creation of node
set n1 [$ns node]
$n0 color red
$n1 color blue
$ns duplex-link $n0 $n1 5Mb 10ms Drop Tail # creating link between the node
$ns duplex-link-op $n0 $n1 orient Right
set Number Flow 200
$ns queue-limit $n0 $n1 20
set tcpscr [new Agent/TCP] #setting TCP agent
$ns attach-agent $n0 $tcpscr
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
set ftp [$tcpscr attach-source/FTP]
set rng1 [new RNG]
$rng seed 0
sey rng2 [new RNG]
$rng2 seed 0
set RV [new RandomVariable/Exponential]
$RVSize set avg_0.045
$RVSize use.rng $rng1
set RVSize [new RandomVariable/Pareto]
$RVSize set avg_10000
$RVSize set shape_1.5
$RVSize use.rng $rng2
set t [$ns now]
$ftp set PacketSize 1000
$ftp set Interval 5
$ns connect $tcpscr $sink
$tcpscr set window_2000
Proc finish{}{
global ns nf qsize
$ns flush-trace
close $qsize
exec xgraph queueSize.tr.geometry 800*400.t “queue size”_x”sec”_y”#packet” & exit
0}
set qfile [$ns monitor-queue $no $n1 [open queue.tr w] 0..5]
#[$ns link $n0 $n1]queue.sample.timeout;
Proc record{}{
global ns qfile qsize n0 n1
set time 0.05
set now [$ns now]
$qfile instvar parrival_pdeparture_bdrops_bdepartures_pdrops_
puts $qsize “$now [ expr $Parrival_$Pdaparture_$Pdrops]
set Pdepartures_0
```

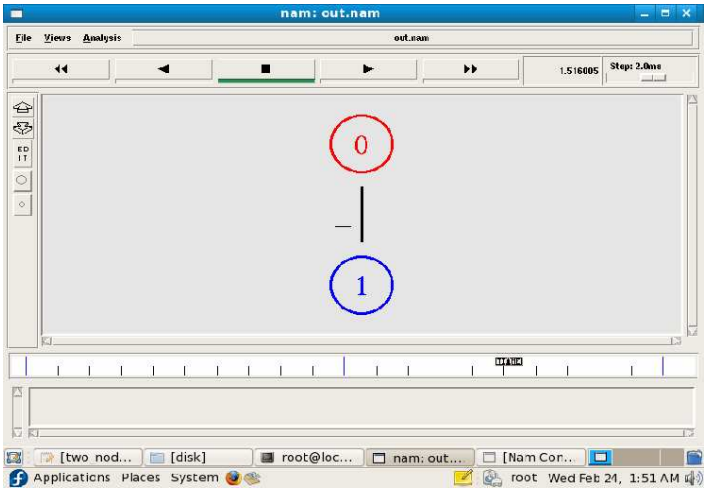
```
$ns at [expr $now+$time]”record
        }
$ns at 0.0 “record”
$ns at 0.1 “$ftp start”
$ns at 4 “$ftp stop”
$ns at 5 “finish”
$ns run
```

4. Result and Analysis: First we have created two nodes. And after simulation trace file and calculation of

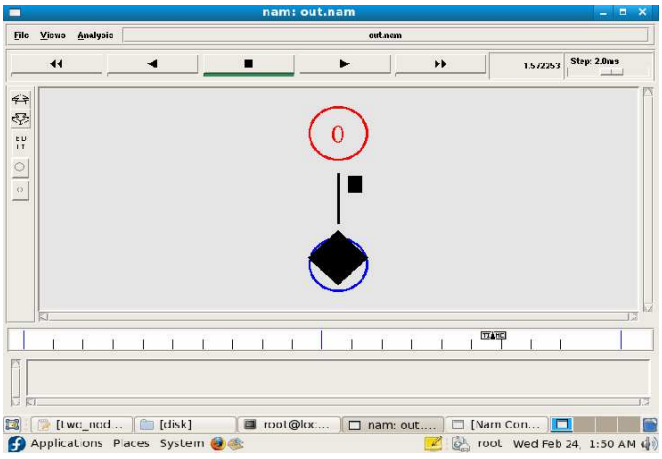
Delay - the average amount of time it takes an access point to start receiving a multicast transmission starting with the “join-group” being sent.

Packet loss – the average amount of data lost in a move between access points. (for a CBR source this is very similar to delay measurements) .

Overhead - the percentage of control messages out of overall traffic on the lines (i.e. control bits/overall bits)[3].



From the above figure we deduce that two node has been created and a link is established between them.



In the above figure we can see packets movement between two nodes.

Above figure is the result of the simulation which shows the graph of queue size in which packets are increased gradually up to 40000 secs, and become constant from 40000 secs onwards. Result data can be seen in tracefile named queuesize.tr.

5. Conclusion: *ns-2* is a popular simulator among the researchers for analyzing the performance of protocols both in wired and wireless networks. This paper describes the simulator in brief and presents an example script for two node network. The two dimensional graph drawn using xgraph utility of the simulator shows the number of packets in queue with time. Similarly, many parameters like packet transfer between nodes, packet loss, throughput, end to end delay etc. can be seen graphically. We can also increase or decrease the data rate, step size, number of nodes etc. to know the performance of network in different conditions.

6. References:

1. http://www.isi.edu/nsnam/ns/tutorial/index.html # formely ns manual
2. Ns Simulator for beginners Lectures notes,Univ de Los Andes, Merida , Venezuela and ESSI, Sophia-Antipolis, France.
3. http://comnet.technion.ac.il/~cn02w04/
4. Marc Greis's tutorial maintained by VINT group.
5. http://en.wikipedia.org/wiki/Ns_simulator
6. T. Issariyakul and E.Hossain, *Introduction to Network Simulator ns2,* Springer.