COMPUTER NETWORKS (CS F303)
SECOND SEMESTER 2018-19
LAB-SHEET – 6
TOPIC: Network simulation using ns2

Learning Objectives:

1. **To learn about network simulations using ns2**
2. **To script and analyze simple network topologies using ns2**
3. **To visualize simple topologies using nam and xgraph**

What is ns2

ns2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks.

Prominent features of ns2

1. It is a discrete event simulator for networking research.
2. It provides substantial support to simulate many protocols like TCP, FTP, UDP and HTTP.
3. It simulates wired as well as wireless networks.
4. Uses TCL and Otcl as its scripting languages. (Otcl provides Object oriented support)
5. It is a discrete event scheduler

Introduction to ns2

The ns2 simulator consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events. The C++ and the OTcl are linked together using TclCL.

Animation and plotting of graphs is performed by separate programs: nam and Xgraph (respectively).

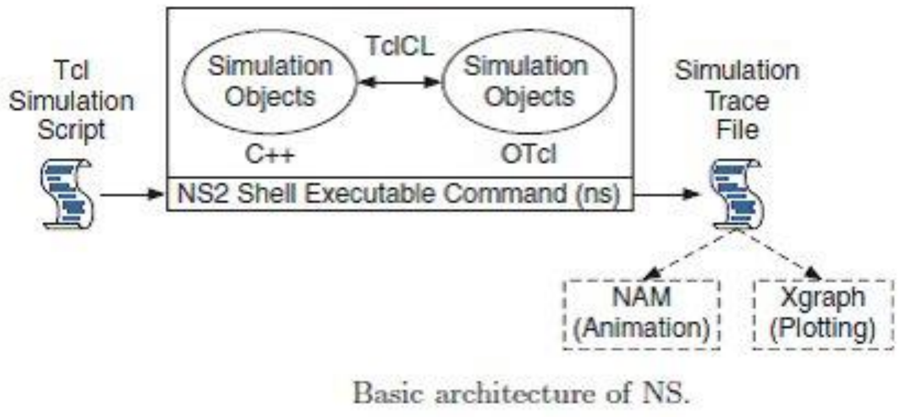

Fig.1

Different components of Tcl/Otcl scripts:

- **The *set* command** :

  ```
  set a 8
  set b [expr $a/8]
  ```

  Here, variable **a** is assigned the value 8. Next, the result of the command [expr $a/8], which equals 1, is used as an argument to another command, which in turn assigns a value to the variable **b**. The "$" sign is used to obtain a value contained in a variable and square brackets are an indication of a command substitution.

- Open a file for writing that is going to be used for the nam trace data (we will discuss more about nam shortly).

  ```
  set nf [open out.nam w]
  $ns namtrace-all $nf
  ```

  We open the file 'out.nam' for writing and give it the file handle 'nf'. Secondly, we tell the simulator object that we created above to write all simulation data that is going to be relevant for nam into this file.

- **Creating the simulator object:**

  The simulator object has member functions which enables to create the nodes and define the links between them. The class simulator contains all the basic functions. Since ns was defined to handle the Simulator object, the command $ns is used for using the functions belonging to the simulator class.

- **Creating nodes and links**

  Nodes can be added in the following manner:

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

  Connecting the two nodes.

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

  This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a DropTail queue.

- **Defining new procedures**

  Add a 'finish' procedure that closes the trace file and starts nam.

  ```
  proc finish {} {
          global ns nf
          $ns flush-trace
          close $nf
          exec nam out.nam &
          exit 0
  }
  ```

- **Creating traffic agents and sources**
  Traffic agents (TCP, UDP etc.) and traffic sources (FTP, CBR etc.) must be set up if the node is not a router. It enables us to create CBR traffic source using UDP as transport protocol or an FTP traffic source using TCP as a transport protocol.

  ```
  #Create a UDP agent and attach it to node n0
  set udp0 [new Agent/UDP]
  $ns attach-agent $n0 $udp0
  # Create a CBR traffic source and attach it to udp0
  set cbr0 [new Application/Traffic/CBR]
  $cbr0 set packetSize_ 500
  $cbr0 set interval_ 0.005
  $cbr0 attach-agent $udp0
  ```

  These lines create a UDP agent and attach it to the node n0, then attach a CBR traffic generator to the UDP agent. CBR stands for 'constant bit rate'. The packetSize is being set to 500 bytes and a packet will be sent every 0.005 seconds (i.e. 200 packets per second).

  Next, we create a Null agent which acts as traffic sink and attach it to node n1.

  ```
  set null0 [new Agent/Null]
  $ns attach-agent $n1 $null0
  ```

  Now the two agents have to be connected with each other.

  ```
  $ns connect $udp0 $null0
  ```

  Finally, we need to tell the CBR agent when to start sending data and when to stop sending, and when the simulator object should execute the 'finish' procedure.

  ```
  $ns at 0.5 "$cbr0 start"
  $ns at 4.5 "$cbr0 stop"
  $ns at 5.0 "finish"
  ```

  Lastly, start the simulation.

  ```
  $ns run
  ```

  Alternatively, FTP/TCP traffic can be generated with a similar approach:

  ```
  set tcp0 [new Agent/TCP]
  $ns attach-agent $n0 $tcp0
  set ftp0 [new Application/FTP]
  $ftp0 attach-agent $tcp0
  $tcp0 set packet_size_ 512
  ```

- Running the script: In your home directory, change to the ns 2.35 directory
  $ cd ns-allinone-2.35/ns-2.35

  Now, you can enter into ns simply by executing ‘ns’, and you can execute tcl scripts in an interactive mode. However, a far more convenient approach is to write your scripts into a file with .tcl extension and execute it using ns <***filename.tcl***>. We will follow this approach.

You have been provided the file **example.tcl**. Run the script on your terminal by executing the following command (remember to provide the path):

```
$ ns /provide/the/path/to/example.tcl
```

Visualizations using nam and Xgraph

Our script invokes the network animator nam on the nam trace file. A new window will open for nam, which looks like as Fig.2.

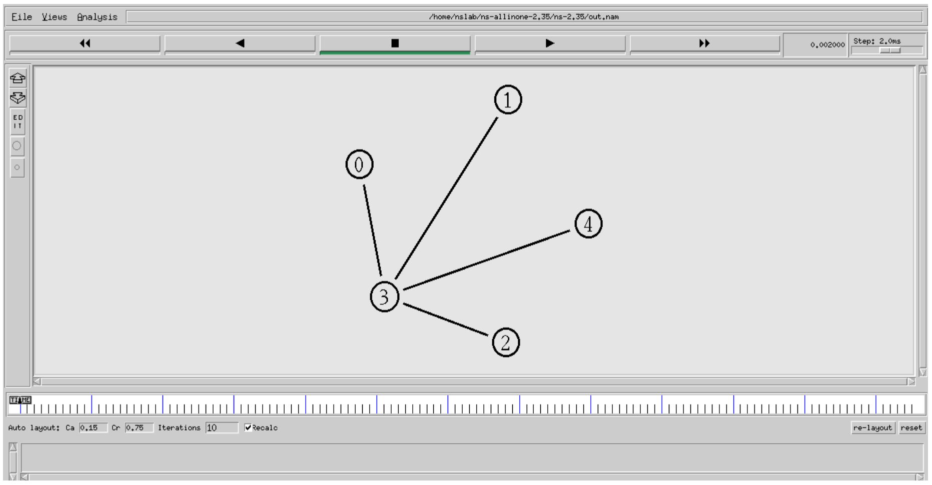
File Views Analysis
/home/nslab/ns-allinone-2.35/ns-2.35/out.nam
0.002000
Step: 2.0ms
0
1
2
3
4
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset

Fig.2

You can click on the play butting to start the animation.

Using nam, you can click on any packet in the window to filter or monitor it, and you can also directly click on the link to get some graphs with statistics.

It is also suggested that you try to change the 'packetsize_' and burst_time_' parameters in the tcl script to see what happens.
Our script also generates a plot using the utility **Xgraph**, which should looks like as shown in Fig.3.

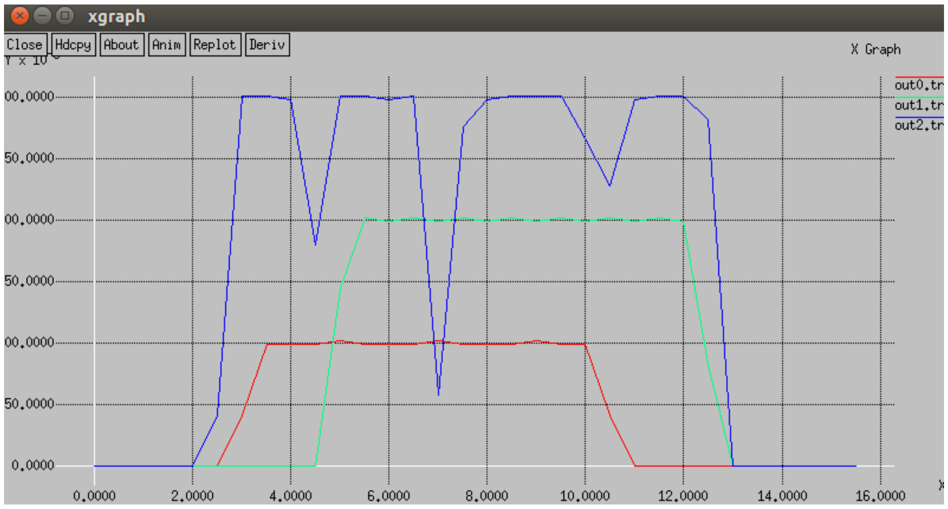

Fig.3

We can observe that the burst of the first flow peak at 0.1Mbit/s, the second at 0.2Mbit/s and the third at 0.3Mbit/s.

Understanding trace files

Simulation results stored in a trace file (outall.tr, in this case). We need to understand the file format for analyzing our results. The format of the trace (.tr) file is given here:

event	time	from node	to node	pkt type	pkt size	flags	flow_id	src addr	Dst addr	seq num	pkt id

Fields of trace file:

- **Event**: type of event i.e., packet dropped, received, added to queue.

Event	Meaning
+	Added to queue
-	Removed from queue
r	received at node
d	dropped

- **Time** – the time in second at which event occurred.
- 3rd and 4th field: “from” and “to” node
- **Packet Type**: Fifth field in trace file represent type of packet. It can be ftp, http, cbr etc. depends on the application attached. Some of them are:

pkt type	Meaning
tcp	TCP packet
cbr	CBR traffic packet
aodv	AODV packet

- **Packet size**: Sixth field show size of packet in bytes.
- **Flags**: There are seven flags which can be used. In case no flags are used, a '-' is included.

Flag	Meaning
-	disabled
C	ECN echo
P	Priority in IP header is enabled
A	Congestion Action
E	Congestion experienced
F	TCP fast start
N	Explicit congestion notification is on

- **Flow id**: this field gives IP flow id as specified in IPv6.
- **Source and Destination address**: Source and destination address is specified in format of "addr.port". Port is also specified with node address.
- **Sequence number**: In ns2, agent can attach sequence number with packets. This field is used only when agent attach sequence number to packets. Although UDP implementations do not use sequence number, ns2 keeps track of UDP packet sequence number for analysis purposes.
- **Packet id**: Every packet has a unique identifier. Every time a new packet is generated, assigned a unique id to it.

A lot of information in trace file might not be of our interest! We can use Linux utilities such as ***grep*** and ***awk*** in order to extract the desired events from the trace file.

Exercises-1

1. What is the kind of link between n0-n3 (or n1-n3)? What other kind of links are possible?
2. What is the effect of increasing (/decreasing) the 'burst_time_' parameter in Expoo traffic agent?
3. What is the effect of increasing (/decreasing) the 'packetSize' parameter in Expoo traffic agent?
4. What does the line 'exec nam out.nam &' do?
5. What is the purpose of creating traffic sinks?
6. In the 'record' procedure, what is the effect of modifying the 'time' procedure to, say, 0.1, 1 or 2? Try these changes and run the simulation again, and then report your observations using Xgraph.
7. Use grep or awk to extract the following information from outall.tr:
    a. The number of 'receive' (r) events between node 0 and node 3.
    b. The number of events apart from 'receive' events between node 1 and node 3.
    c. The number of "added to queue" events originating from node 3. How many of such events were to a node apart from node 4?

Exercises-2

Write a tcl script to simulate network topology shown in Fig. 4:

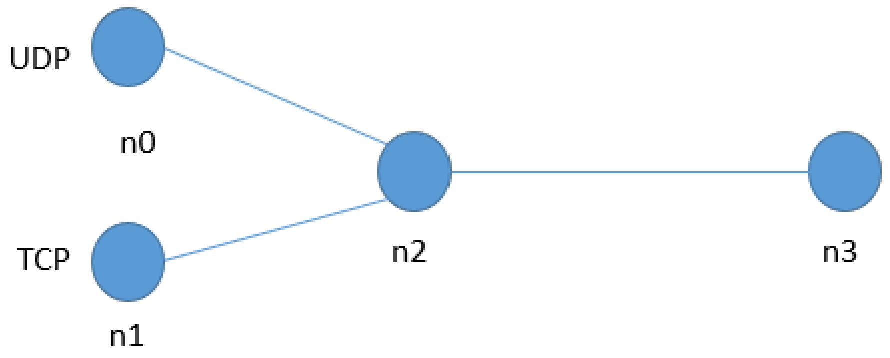

Fig.4

Node n0 communicates with n3 over UDP, while node n1 communicates with n3 over TCP. All links 2Mb with 10ms delay. For UDP, you may use a CBR traffic source with packet size of 500 bytes and interval parameter of 0.005. For TCP, you may use a FTP traffic source.

Start the simulation scenario with only TCP traffic. Start the UDP traffic after 5 seconds. End all traffic and finish the simulation at 10 seconds.

Tasks:

a. Find out the number of UDP packets dropped with respect to time.
b. Plot the variation in latency of packets (i.e., jitter) received at node n3 with respect to time for TCP flow.
c. Now modify your script such that UDP flow sending rate becomes double. Run the simulation and plot the graphs for jitter and dropped packets.
d. Calculate the throughput for TCP and UDP flows in both cases.

References:

1. https://www.isi.edu/nsnam/ns/tutorial/
2. https://www.geeksforgeeks.org/basics-of-ns2-and-otcltcl-script/
3. https://www.tutorialsweb.com/ns2/index.htm

******