NETWORK SIMULATOR-2

(NS2)

TEAM MEMBERS:

SONU KUMAR

MEHGNA KM

POOJA M

History

ns-1

The first version of ns, known as ns-1, was developed at Lawrence Berkeley National Laboratory (LBNL) in the 1995-97 timeframe by Steve McCanne, Sally Floyd, Kevin Fall, and other contributors. This was known as the LBNL Network Simulator, and derived from an earlier simulator known as REAL by S. Keshav.

ns-2

Ns began as a variant of the REAL network simulator in 1989 and has evolved substantially over the past few years. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. Currently ns development is support through DARPA with SAMAN and through NSF with CONSER, both in collaboration with other researchers including ACIRI. Ns has always included substantial contributionswireless code from the UCB Daedelus and CMU Monarch projects and Sun Microsystems. For documentation on recent changes, see the version 2 change log.

ns-3

A team led by Tom Henderson, George Riley, Sally Floyd, and Sumit Roy, applied for and received funding from the U.S. National Science Foundation (NSF) to build a replacement for ns-2, called ns-3. This team collaborated with the Planete project of INRIA at Sophia Antipolis, with Mathieu Lacage as the software lead, and formed a new open source project.

In the process of developing ns-3, it was decided to completely abandon backward-compatibility with ns-2. The new simulator would be written from scratch, using the C++ programming language. Development of ns-3 began in July 2006.

The first release, ns-3.1 was made in June 2008, and afterwards the project continued making quarterly software releases, and more recently has moved to three releases per year. ns-3 made its twenty first release (ns-3.21) in September 2014.

NOTE:

Though ns-3 is currently being actively developed ,but your area of interest is ns-2 and we'll discuss the same.

INTRODUCTION

NS-2 is an event driven packet level network simulator developed as part of the VINT project (Virtual Internet Testbed). This was a collaboration of many institutes including UC Berkeley, AT&T, XEROX PARC and ETH. Version 1 of NS was developed in 1995 and with version 2 released in 1996. Version 2 included a scripting language called Object oriented Tcl (OTcl). It is an open source software package available for both Windows and Linux platforms. It provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks.

NS-2 has many and expanding uses including:

- To evaluate the performance of existing network protocols.
- To evaluate new network protocols before use.
- To run large scale experiments not possible in real experiments.
- To simulate a variety of ip networks

STRUCTURE OF NS2

a. NS is an object oriented discrete event simulator

i. Simulator maintains list of events and executes one event after another.

ii. Single thread of control: no locking or race conditions.

b. Back end is C++ event scheduler.

i.Protocols mostly.

c. Source code:

i.Most of process procedures of NS2 code are written in C++ code.

d. Scripting language:

i. It uses TCL as its scripting language OTcl adds object Orientation to TCL.

e. Protocols implemented in NS2:

i. Transport layer (Traffic Agent) – TCP, UDP.(TCP using in our design of wireless network).

ii. Interface queue, Drop Tail queue.

f. Scalability:

i. Per-packet processing must be fast;

ii. Separating control and packet handling.

g. Import C++ code to TCL script program

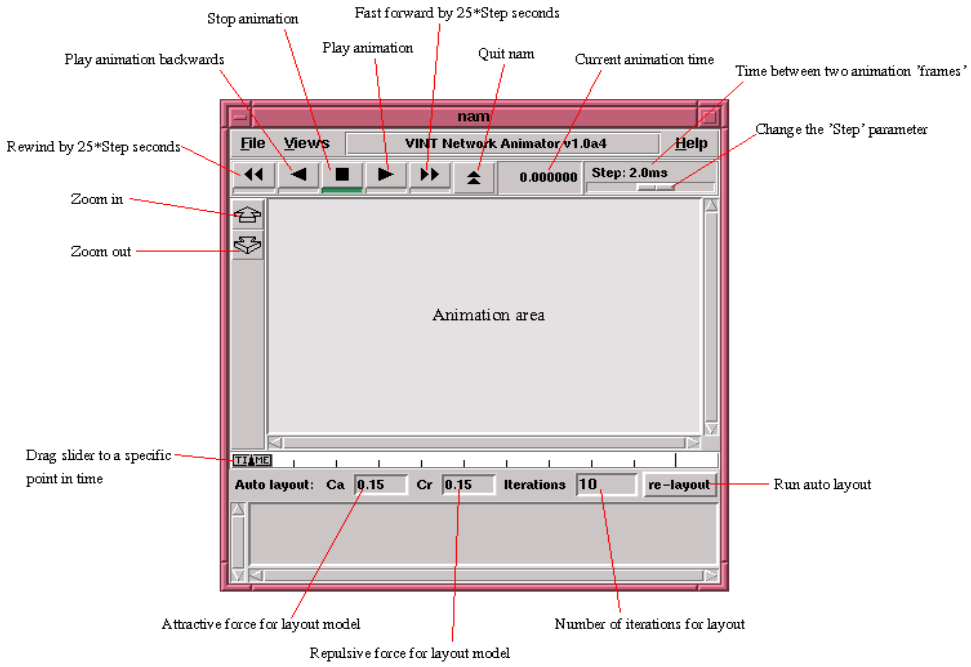
Simulation workflow

The general process of creating a simulation can be divided into several steps:

1. ***Topology definition*:** To ease the creation of basic facilities and define their interrelationships, ns-3 has a system of containers and helpers that facilitates this process.
2. ***Model development*:** Models are added to simulation (for example, UDP, IPv4, point-to-point devices and links, applications); most of the time this is done using helpers.
3. ***Node and link configuration*:** models set their default values (for example, the size of packets sent by an application or MTU of a point-to-point link); most of the time this is done using the attribute system.
4. ***Execution*:** Simulation facilities generate events, data requested by the user is logged.
5. ***Performance analysis*:** After the simulation is finished and data is available as a time-stamped event trace. This data can then be statistically analysed with tools like R to draw conclusions.
6. ***Graphical Visualization*:** Raw or processed data collected in a simulation can be graphed using tools like Gnuplot, matplotlib or XGRAPH.

NAM (Network Animator)

NAM provides a visual interpretation of the network topology created. Below you can see a screenshot of a nam window where the most important functions are being explained.



Its features are as follows:

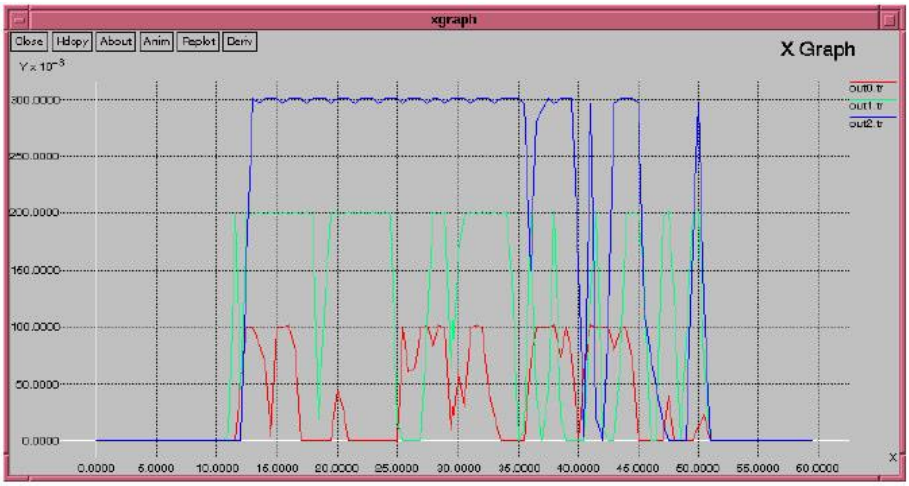
- Provides a visual interpretation of the network created
- Can be executed directly from a Tcl script
- Controls include play, stop ff, rw, pause, a display speed controller and a packet monitor facility.
- Presents information such as throughput, number packets on each link.
- Provides a drag and drop interface for creating topologies.

1.8 XGraph

XGraph is an X-Windows application that includes:

- Interactive plotting and graphing
- Animation and derivatives

To use XGraph in NS-2 the executable can be called within a TCL Script. This will then load a graph displaying the information visually displaying the information of the trace file produced from the simulation.



XGraph running comparing three trace files in a graph

1.9 TraceGraph

TraceGraph is a trace file analyser that runs under Windows, Linux and UNIX systems and requires Matlab 6.0 or higher.

TraceGraph supports the following trace file formats.

- Wired
- Satellite
- Wireless

The problem statement should be addressed and a solution should be formulate for the same.

Problem Statement:

Simulate a four node point-to-point network, and connect the links as follows: n0-n2, n1-n2 and n2-n3. Apply TCP agent between n0-n3 and UDP n1-n3. Apply relevant

applications over TCP and UDP agents changing the parameter and determine the number of packets by TCP/UDP.

Solution:

```
set ns [new Simulator]

#define different colors for data flow

$ns color 1 Blue

$ns color 2 Red

#Open a new file for NAMTRACE

set nf [open out.nam w]

$ns namtrace-all $nf



#Open a new file to log TRACE

set tf [open out.tr w]

$ns trace-all $tf



#Body of the 'finish' procedure

proc finish {} {

        global ns nf tf

        $ns flush-trace

        close $nf
```

```
        close $tf
            exec nam out.nam &
            exit 0
}

#Create Nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create Links between Nodes
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail

#Set the queue limit - default is 50 packets
$ns queue-limit $n0 $n2 50
$ns queue-limit $n1 $n2 50
$ns queue-limit $n2 $n3 50
```

```
#Create TCP Agent between node 0 and node 3

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink0 [new Agent/TCPSink]

$ns attach-agent $n3 $sink0

$ns connect $tcp0 $sink0

$tcp0 set class_1


#Create FTP Application for TCP Agent

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0


#Specify TCP packet size

Agent/TCP set packetSize_ 1000


#Create UDP Agent between node 1 and node 3

set udp0 [new Agent/UDP]

$ns attach-agent $n1 $udp0

set null0 [new Agent/Null]
```

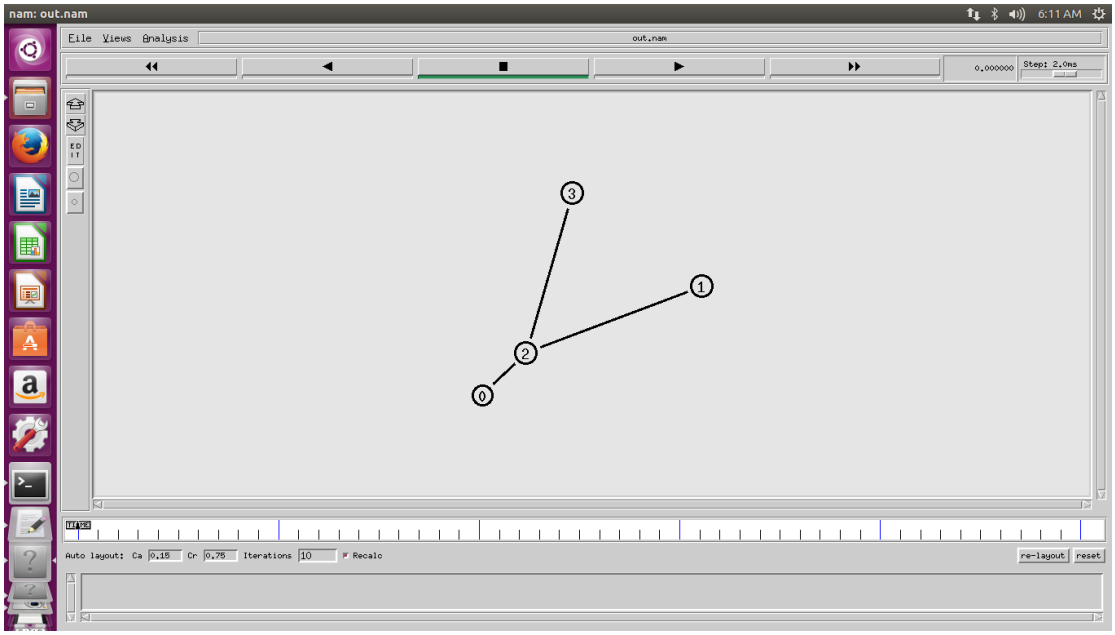
```
$ns attach-agent $n3 $null0

$ns connect $udp0 $null0

$udp0 set class_2



#Create CBR Application for UDP Agent

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0


#Start and Stop FTP Traffic

$ns at 0.75 "$ftp0 start"

$ns at 4.75 "$ftp0 stop"


#Start and Stop CBR traffic

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"


#Stop the simulation
```

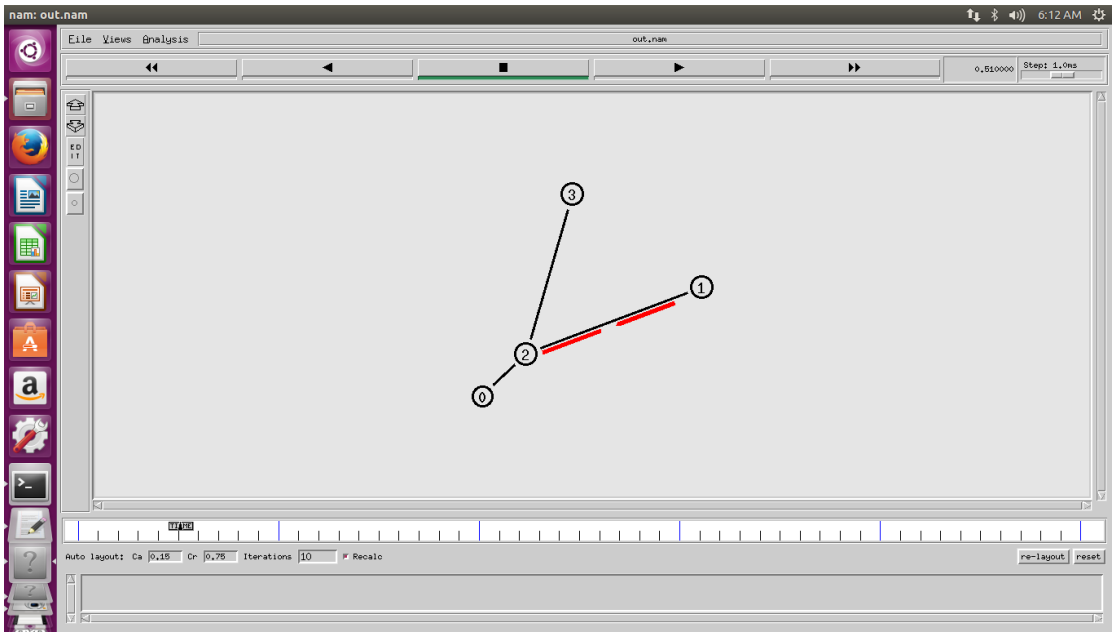
$ns at 5.0 "finish"

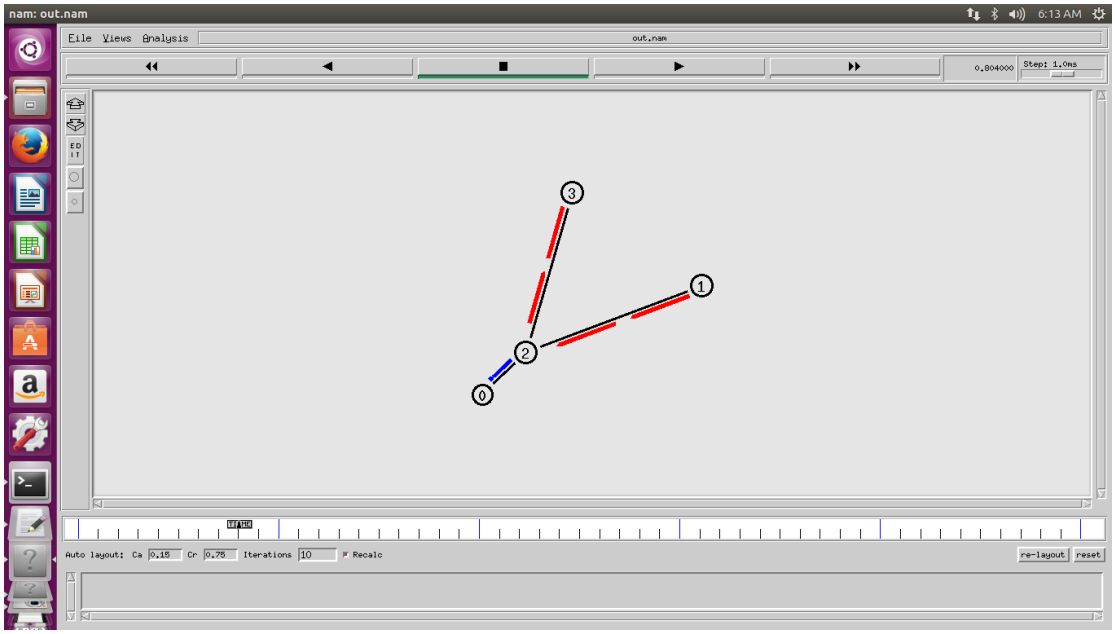
#Run the simulation

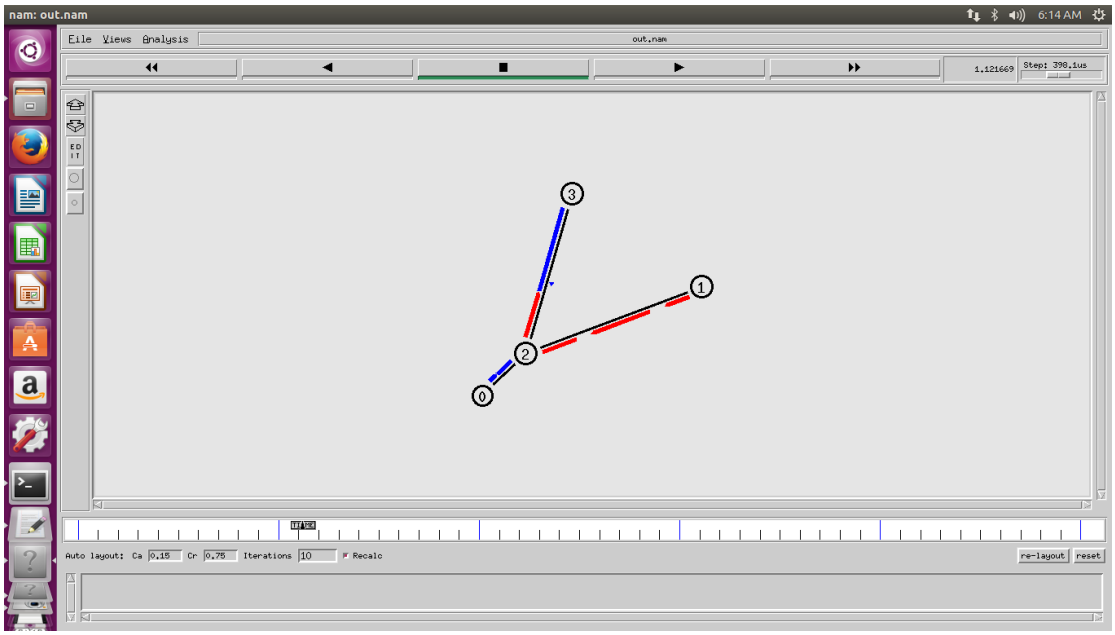
$ns run

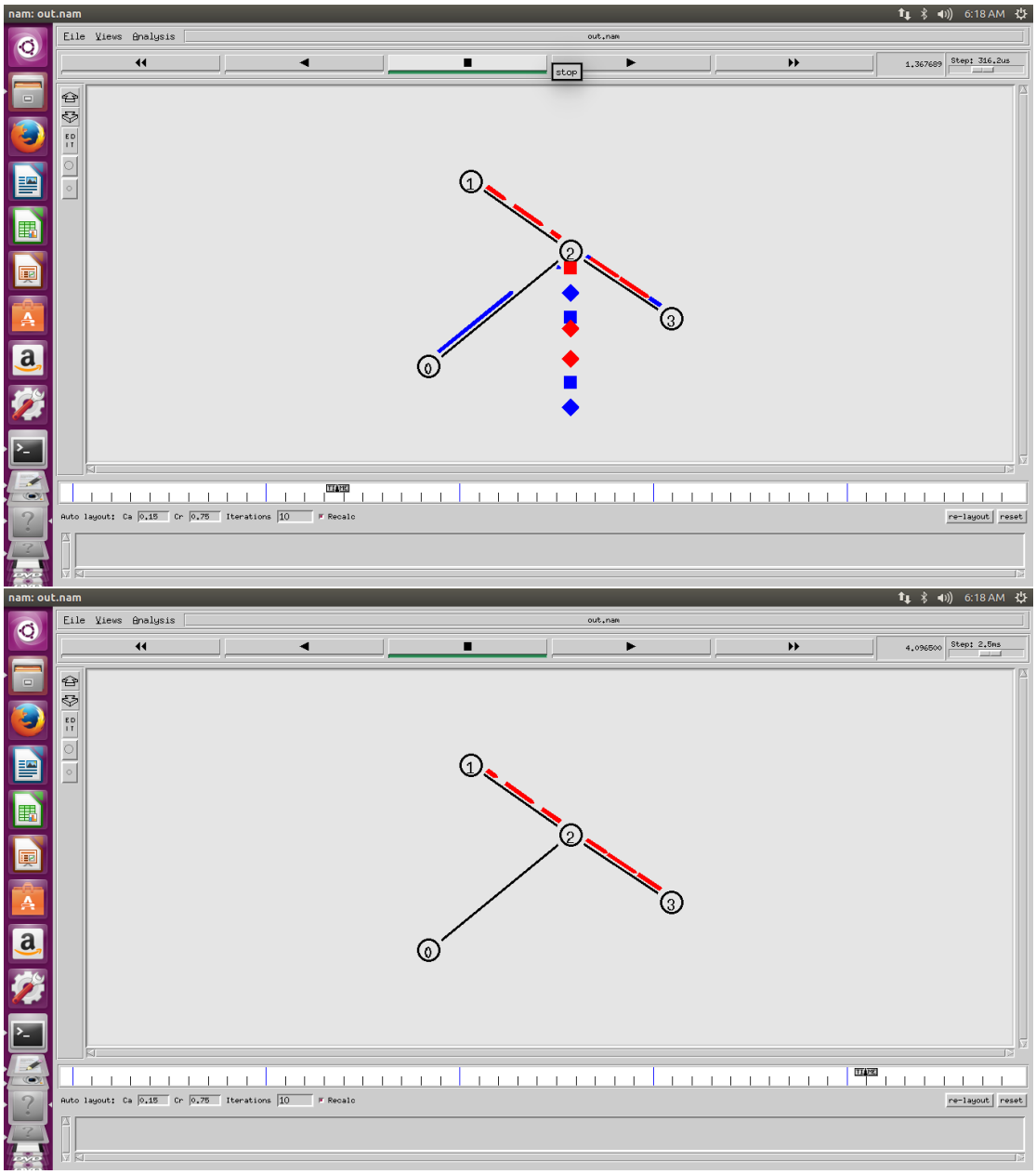
Screenshot

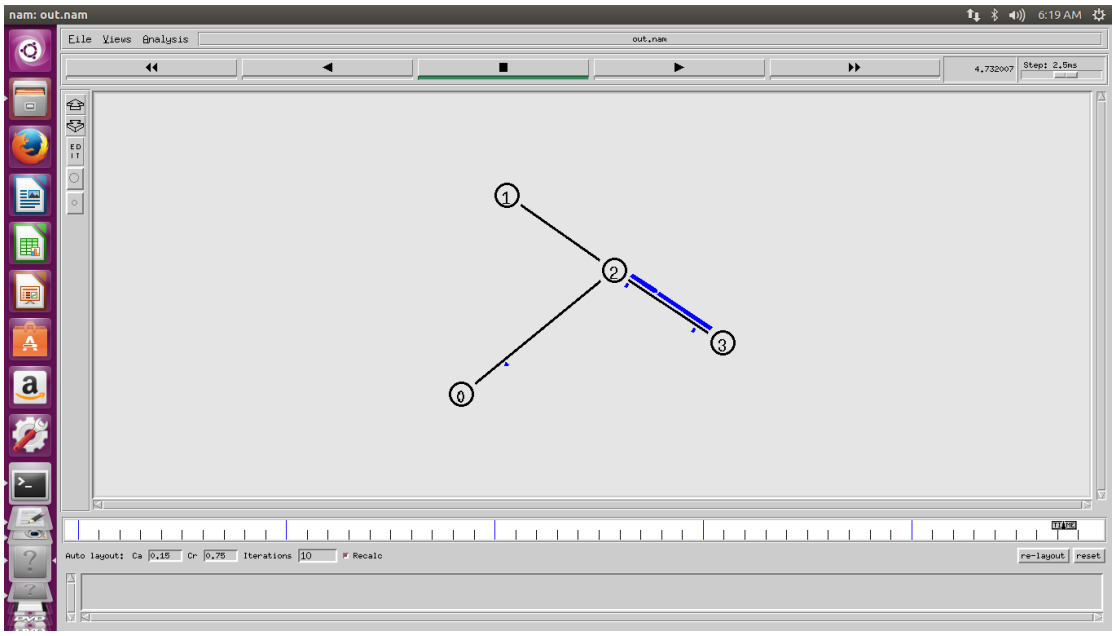





nam: out.nam
6:13 AM
File Views Analysis
out.nam
0.804000
Step: 1.0ms
3
1
2
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset


nam: out.nam
6:14 AM
File Views Analysis
out.nam
1.121669
Step: 390.1us
3
1
2
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset


nam: out.nam
6:18 AM
File Views Analysis
out.nam
stop
1.367689
Step: 316.2us
1
2
3
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset
nam: out.nam
6:18 AM
File Views Analysis
out.nam
4.096500
Step: 2.5ms
1
2
3
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout reset


nam: out.nam
6:19 AM
File Views Analysis
out.nam
4.732007
Step: 2.5ms
1
2
3
0
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc
re-layout
reset