|  |  |
| --- | --- |
| **Course Name: Computer Networks** | **Course Code: CMPE-333L** |
| **Assignment Type:** Lab | **Dated: 14-04-2024** |
| **Semester: 6th** | **Session:** 2021-2025 |
| **Lab #: 8** | **CLOs to be covered: 3** |
| **Lab Title: Network Simulations (Using NS2)** | **Teacher Name: Darakhshan Abdul Ghaffar** |
| **Student Name: Urwah Imran** | **Student Roll No: 2021-CE-15** |

**Lab Evaluations:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CLO3** | Build simulations to measure the network parameters | | | |  |  |
| **Levels (Marks)** | **Level1** | **Level2** | **Level3** | **Level4** | **Level5** | **Level6** |
| Completeness  (5) |  |  |  |  |  |  |
|  |  | | | | **Total** | **/10** |

**Rubrics for Current Lab (Optional):**

|  |  |  |  |
| --- | --- | --- | --- |
| **Scale** | **Marks** | **Level** | **Rubric** |
| **Excellent** | **5** | L1 | Submitted all lab tasks during the lab, have good understanding. |
| **Very Good** | **4** | L2 | Submitted the lab tasks but have weak understanding |
| **Good** | **3** | L3 | Submitted the lab tasks but have weak understanding. |
| **Basic** | **2** | L4 | Submitted the lab tasks but have no understanding. |
| **Barely**  **Acceptable** | **1** | L5 | Submitted only one lab task. |
| **Not**  **Acceptable** | **0** | L6 | Lab missed or implemented none of the task |

**LAB DETAILS:**

**Lab Goals/Objectives:**

• To understand the network simulation

**Equipment Required:** Computer system with NS2 simulator installed on it an online NS2 simulator.

**Theory/Relevant Material:**

# What is NS

* Started as REAL in 1989
* Discrete event, packet level simulator
* Written in C++ with Otcl frontend
* Wired, wireless and emulation modes
* Link layer and up for most wired

# Platforms

* Most UNIX and UNIX-like systems
* Linux
* FreeBSD
* SunOS/Solaris
* HP/SGI (with some tweaking)
* Windows 95/98/NT/ME/2000  Tips on build available
* However validation tests don’t work

# Architecture of NS

* Object-oriented (C++ and Otcl) o Algorithms over large data sets, per packet handling in C++ o Configuration, “one-time” stuff in Otcl o Fast to run, quick to re-configure
* Fine grained object composition
* C++ for “data”
* Per packet action  OTcl for control
* Configuration, “one-time” task
* Compromise between composibility and speed

|  |  |
| --- | --- |
|   **Basic TCL** | Learning and debugging  set a 43 set b 27 proc test { a b } { set c [expr $a + $b] set d [expr [expr $a - $b] \* $c] for {set k 0} {$k < 10} {incr k} { if {$k < 5} { puts “k < 5, pow = [expr pow($d, $k)]”  } else { puts “k >= 5, mod = [expr $d % $k]”  }  } |
| **Basic OTCL**  Class Mom  Mom instproc greet {} { $self instvar age\_  puts “$age\_ years old mom: How are you doing?”  }  Class Kid -superclass Mom  Kid instproc greet {} { $self instvar age\_  puts “$age\_ years old kid: What’s up, dude?” **set** mom [**new** Mom] | |

$mom **set** age\_ 45 **set** kid [**new** Kid] $kid **set** age\_ 15

$mom greet

$kid greet

# Hello World in NS

%ns

% set ns [new Simulator]

% $ns at 1 “puts \“Hello World!\””

% $ns at 1.5 “exit”

% $ns run

# Event Driven Simulation

* Scheduler – main controller of events
* Scheduler clock - simulator virtual time
* [$ns\_ now] returns the current simulator time
* Event queue - holds events in the order of their firing times
* Events have a firing time and a handler function
* Two types of events possible – packets and “at-events” **Discrete Event Scheduler**



time\_, uid\_, next\_, handler\_



Head\_

-

>



handler\_

-

()

handle

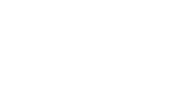
>



time\_, uid\_, next\_, handler\_



Insert



Event

Queue



Deque



Dispatch



Reschedule

Four types of scheduler:

* + List: simple linked list, order-preserving, O(N)
  + Heap: O(logN)
  + Calendar: hash-based, fastest, default, O(1)
  + Real-time: subclass of list, sync with real-time, O(N)

# Procedure of NS

* Create the event scheduler
* [Turn on tracing]
* Create network
* Setup routing
* Insert errors
* Create transport connection
* Create traffic

# Create an Event Scheduler

* Create event scheduler
* set ns [new Simulator]  Schedule events
* $ns at <time> <event>
* <event>: any legitimate ns/tcl commands  Start scheduler
* $ns run

# Tracing

* Trace packets on all links

– $ns trace-all [open test.out w]

* + <event> <time> <from> <to> <pkt> <size> -- <fid> <src> <dst> <seq> <attr>
  + + 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
* Trace packets on all links in nam format
  + $ns namtrace-all [open test.nam w]
  + $ns namtrace-all-wireless [open wtest.nam w]
* Turn on tracing on specific links
  + $ns trace-queue $n0 $n1
  + $ns namtrace-queue $n0 $n1
  + Trace-all commands must appear immediately after creating scheduler  Event tracing
  + $ns eventtrace-all [$file]
  + Add eventtrace *after* trace-all as trace-all file is used as default
  + Example script: ~ns/tcl/ex/tcp-et.tcl

# Creating NS Topology

* Nodes
  + set n0 [$ns node] o set n1 [$ns node]
* Links and queuing
  + $ns duplex-link $n0 $n1 <bandwidth> <delay> <queue\_type> o <queue\_type>: DropTail, RED, CBQ, FQ, SFQ, DRR o $ns duplex-link $n0 $n1 5Mb 2ms DropTail

# Inserting Errors

* Creating Error Module o set loss\_module [new ErrorModel] o $loss\_module set rate\_ 0.01 o $loss\_module unit pkt o $loss\_module ranvar [new RandomVariable/Uniform] o $loss\_module drop-target [new Agent/Null]
* Inserting Error Module o $ns lossmodel $loss\_module $n0 $n1

# Network Dynamics

* Link failures o Hooks in routing module to reflect routing changes
* Four models o $ns rtmodel Trace <config\_file> $n0 $n1 o $ns rtmodel Exponential {<params>} $n0 $n1 o $ns rtmodel Deterministic {<params>} $n0 $n1 o $ns rtmodel-at <time> up|down $n0 $n1
* Parameter list o [<start>] <up\_interval> <down\_interval> [<finish>]

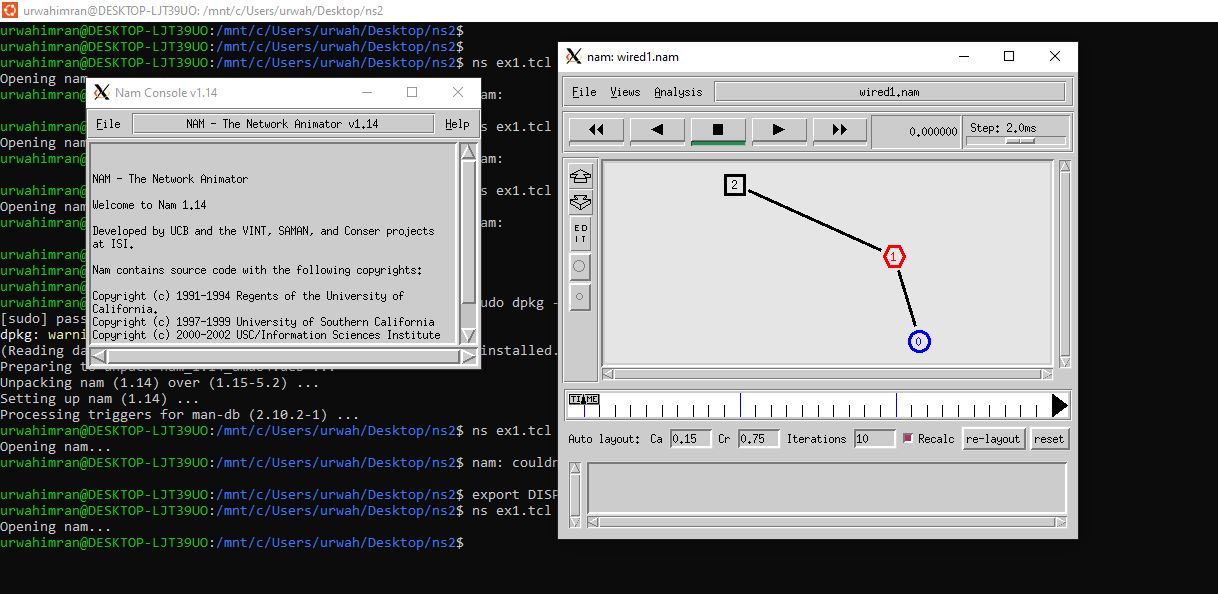
# Setup Routing

* Unicast o $ns rtproto <type>
  + <type>: Static, Session, DV, LS, Manual or hierarchical
* Multicast
  + $ns multicast (right after [new Simulator]) o $ns mrtproto <type>
* <type>: CtrMcast, DM, ST, BST

# Creating TCP Connections

 One-way TCP sending agent [Tahoe, Reno, NewReno, Sack, Vegas and Fack] o set tcp [new Agent/TCP] o set tcpsink [new Agent/TCPSink] o $ns attach-agent $n0 $tcp o $ns attach-agent $n1 $tcpsink o $ns connect $tcp $tcpsink

**Solution: Final Step of ns2 installation**

****

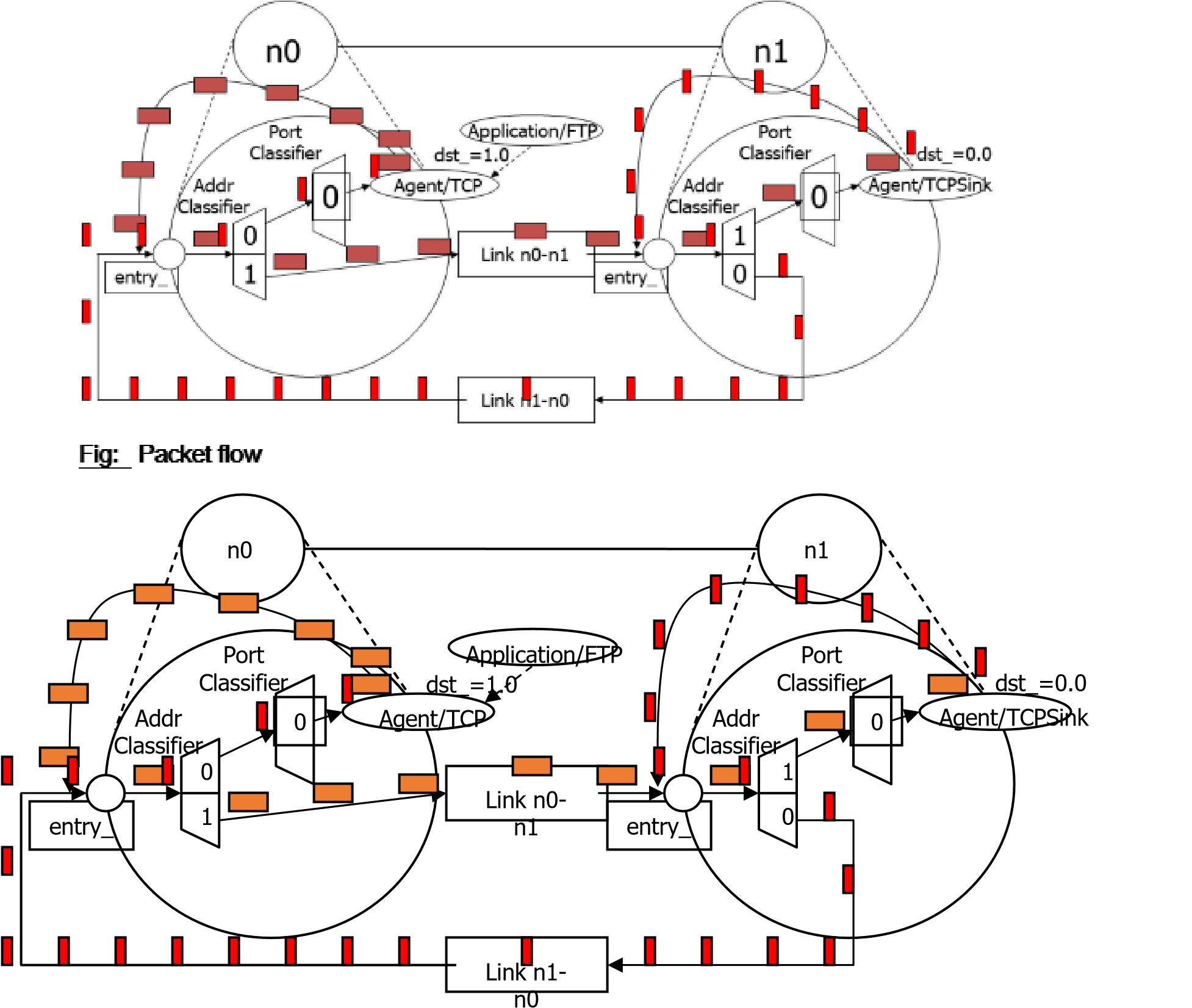
**Lab Tasks:**

# Creating Traffic on the top of TCP

* FTP

o set ftp [new Application/FTP] o $ftp attach-agent $tcp

* Telnet o set telnet [new Application/Telnet] o $telnet attach-agent $tcp



# Fig: Packet flow

**Generic Script Structure Summary** set ns [new Simulator] # [Turn on tracing]

# Create topology

# Setup packet loss, link dynamics # Create routing agents # Create:

# - multicast groups

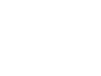
# - protocol agents

# - application and/or setup traffic sources

# Post-processing procs

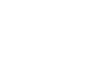
# Start simulation

## Example TCP Network (A Simple Scenario)



n0

TCP



n4

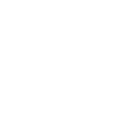
UDP



n1



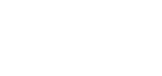
n2



n5

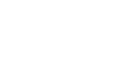
UDP

recvr



n3

TCPSink

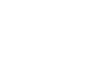


Mb

1.5

10

ms

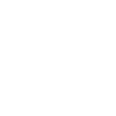


Mb

5

2

ms



5

Mb

2

ms

**Step 1:**

 Scheduler & tracing o Create scheduler

* Set ns [new Simulator] o Turn on tracing
* set f [open out.tr w]
* $ns trace-all $f
* Set nf [open out.nam w]
* $ns namtrace-all $nf

## Step 2

 Create topology o create nodes

* set n0 [$ns node]
* set n1 [$ns node]
* set n3 [$ns node]
* set n4 [$ns node]

## Step 3

 create links

o $ns duplex-link $n0 $n1 5Mb 2ms DropTail o $ns duplex-link $n1 $n2 1.5Mb 10ms DropTail o $ns duplex-link $n2 $n3 5Mb 2ms DropTail o $ns queue-limit $n1 $n2 25 o $ns queue-limit $n2 $n1 25

## Step 4

* Create TCP agents
* set tcp [new Agent/TCP]
* set sink [new Agent/TCPSink]
* $ns attach-agent $n0 $tcp
* $ns attach-agent $n3 $sink
* $ns connect $tcp $sink

## Step 5

* Attach traffic
* set ftp [new Application/FTP]
* $ftp attach-agent $tcp
* #start application traffic
* $ns at 1.1 “$ftp start”

## Step 6

* End of simulation wrapper (as usual)
  + $ns at 2.0 “finish”
  + Proc finish {}

 {

* global ns f nf
* close $f
* close $nf  puts “Running nam…”
* exec nam out.nam &
* exit 0
* } o $ns run

## Visualization Tools

### Nam-1 (Network AniMator Version 1)

* Packet-level animation
* Well-supported by ns Xgraph

 Convert trace output into xgraph format

## NAM interface

* Color
* Node manipulation
* Link manipulation
* Topology layout
* Protocol state
* Misc

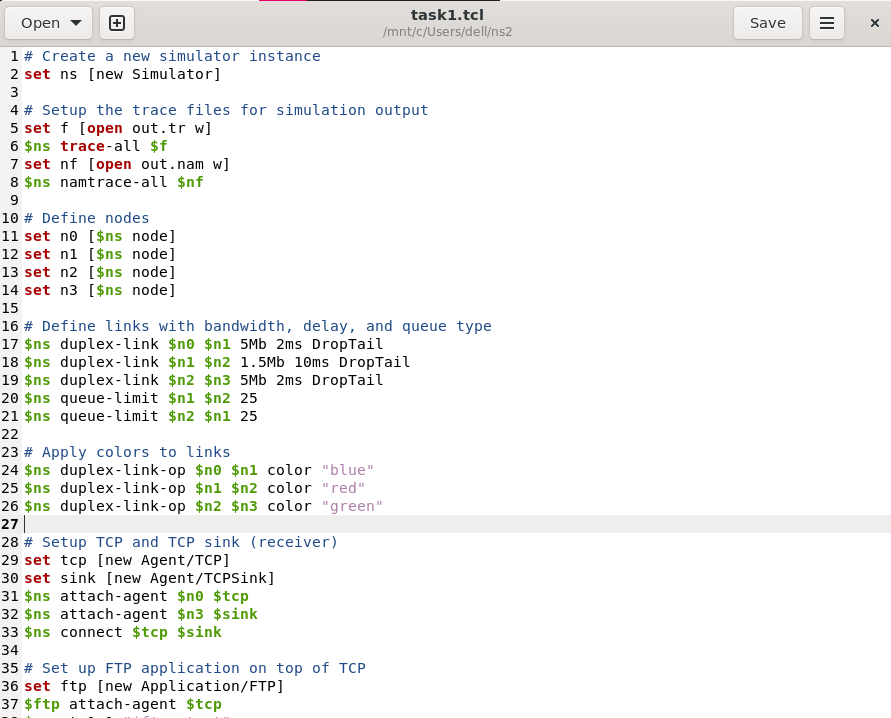
### NAM interface (Color)

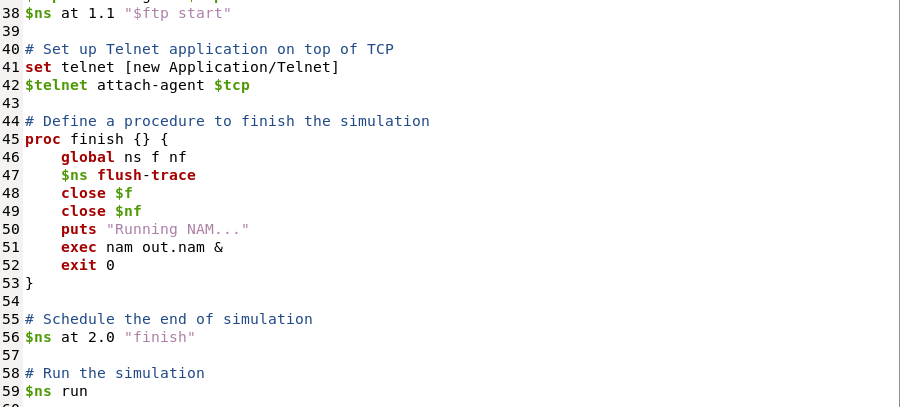
 Color mapping o $ns color 40 red o $ns color 41 blue o $ns color 42 chocolate o Color  flow id association o $tcp0 set fid\_ 40 ;# red packets o $tcp1 set fid\_ 41 ;# blue packets o Color (Nodes)

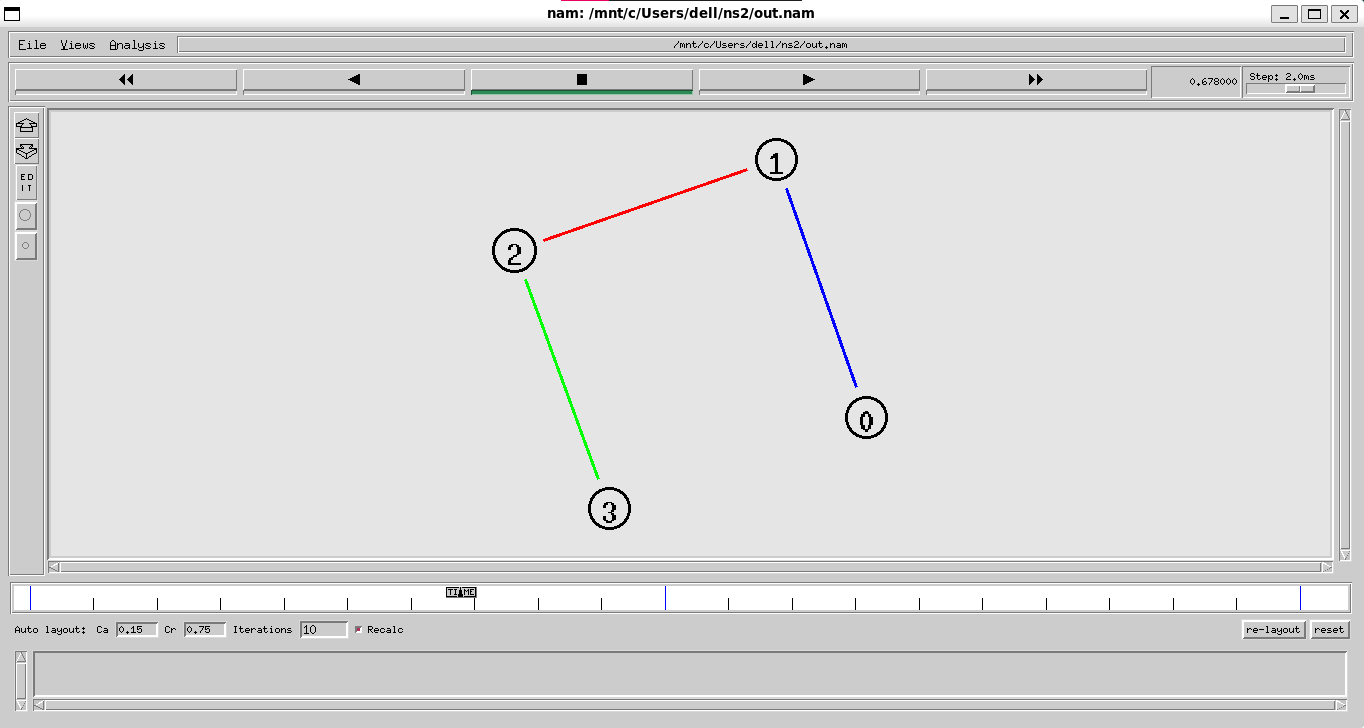
* node color red o Shape (can’t be changed after sim starts)
* $node shape box ;# circle, box, hexagon o Marks (concentric “shapes”)
* $ns at 1.0 “$n0 add-mark m0 blue box”
* $ns at 2.0 “$n0 delete-mark m0” o Label (single string)
* $ns at 1.1 “$n0 label \”web cache 0\”” o Color (links)
* $ns duplex-link-op $n0 $n1 color "green" o Label
* $ns duplex-link-op $n0 $n1 label "abced" o Dynamics (automatically handled)
* $ns rtmodel Deterministic {2.0 0.9 0.1} $n0 $n1 o Asymmetric links not allowed

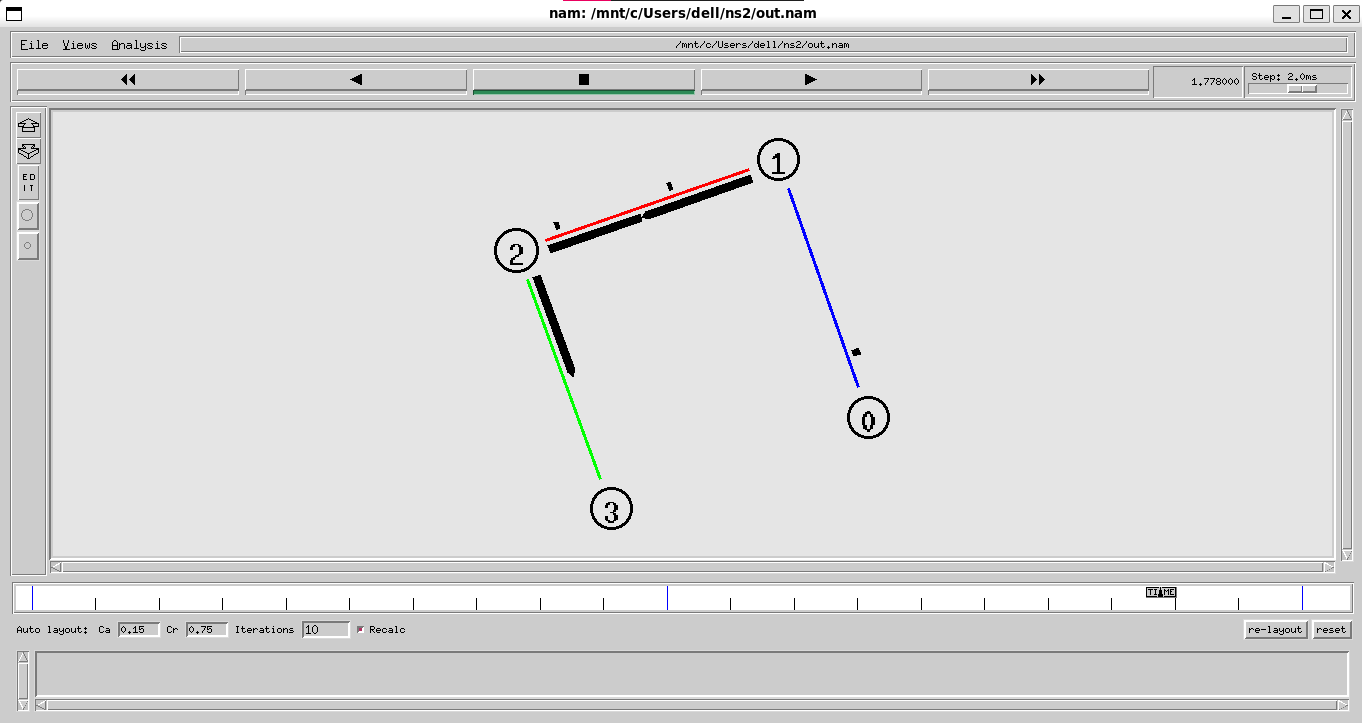
## Topology Layout NAM

* “Manual” layout: specify everything o $ns duplex-link-op $n(0) $n(1) orient right o $ns duplex-link-op $n(1) $n(2) orient right-up o $ns duplex-link-op $n(2) $n(3) orient down o $ns duplex-link-op $n(3) $n(4) orient 60deg
* If anything missing  automatic layout
* **TCL script for Lab Task:**

****

****

****

****