Network Performance and NS2

A tutorial by:

Mohammed M. Kadhum *PhD MIEEE*

InterNetWorks Research Group
Universiti Utara Malaysia
06010 UUM Sintok, MALAYSIA

kadhum@uum.edu.my
kadhum@internetworks.my

Learning Outcomes

At the end of this session, participants should be able to:

- *Explain* and *identify* network performance evaluation techniques commonly used in the computer systems and networking research
- *Describe* the simulation-based network performance evaluation technique commonly used in the computer systems and networking research
- *Utilize* NS2 network simulator in their networking research

Session Outline

- Introduction to network performance evaluation
- Network performance evaluation techniques
  - analytical model
  - measurement
  - simulation
- Network simulation using NS2
- Demos and Discussions

What’s Network Performance Study?

Performance:

- key criterion in the
  - design, and
  - use of computer and communication systems

Performance evaluation:

- to compare a number of alternative designs and finds the best design

Why Performance Evaluation?

- determine performance measures
  - for existing systems or,
  - for models of (existing or future) systems
- develop new analytical and methodological foundations,
  - e.g. in queueing theory, simulation etc.
- find ways to apply and validate theoretical approaches
  - e. g. in creating and evaluating performance models

Performance Evaluation Techniques

Techniques commonly used in the computer systems and networking research

- Measurements
- Analytical Modeling
- Simulation

Measurement

- can be done in a test-bed network or an operational network
- requires real equipment, code and time to run experiments
  - Monitoring
  - Prototyping
    - if the system is new or not available
    - e.g. a new network protocol
- normally used when system under study already exists and is accessible

Measurement: Drawbacks

- Network test beds
  - can be difficult to configure and reconfigure and share among researchers
  - may be too disruptive to do a measurement by testing on real operational networks
  - can be very expensive
  - requires accumulated experience to do the measurement,
    - in particular if need to prototype and build a test-bed network on an interesting scale

Analytical Modeling

- uses mathematical notions and models
    - to describe performance aspects of the system under study
- requires construction of a mathematical model of the system such as using queuing networks and Petri-Nets
- the cheapest and least time-consuming technique compared with measurement or simulation
- analytical modeling results can have better predictive values than measurement or simulation
- a good tool to study overall network characterization

Analytical Modeling: Drawbacks

- generally requires too many simplifications and assumptions
  - … may be inaccurate about the real network
- ignores network dynamics (such as flow interactions)
  - … can prove to be critical in practice
- Most network protocols and systems are too complex to be realistically modeled using analytical modeling
  - e.g.: sequence of events leading to network congestion can be complex and generally hard to analyze analytically

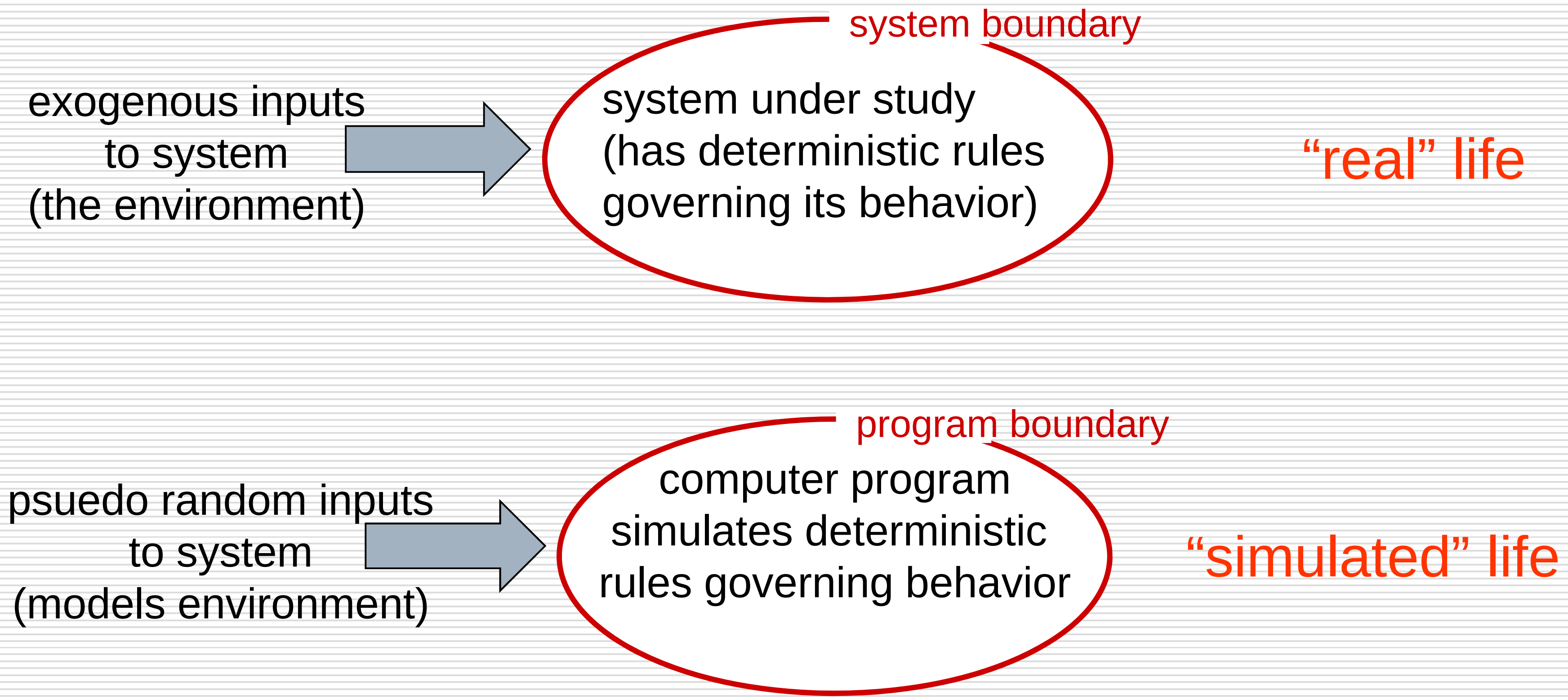
Simulation

- one of the most commonly used paradigms in the study of communication networks
- has been the main research methodology used by researchers working on the core of Internet development
- Simulation usages:
  - study how existing system/network works
  - study non-existing system/network without building it
  - explore proposals in environments that have not been realized in the current Internet but may be in the future
  - characterizing both behavior of the current Internet and the possible effects of proposed changes to its operation

Network Simulation

- has been the main research methodology used by researchers working on the core of Internet development
- number of credible published research works has been done using network simulation
  - e.g. those that appear in IEEE/ACM journals and proceedings
- used to ensure that functional requirements of new network elements (e.g.. new protocols) can be achieved and are feasible

What is simulation? *


system boundary
exogenous inputs to system (the environment)
system under study (has deterministic rules governing its behavior)
“real” life
program boundary
psuedo random inputs to system (models environment)
computer program simulates deterministic rules governing behavior
“simulated” life

Why Simulation? *

- real-system not *available, is complex/costly or dangerous* (eg: space simulations, flight simulations)
- quickly evaluate design *alternatives* (eg: different system configurations)
- evaluate *complex functions* for which closed form formulas or numerical techniques not available

Simulation: advantages/drawbacks*

- advantages:
  - sometimes cheaper
  - find bugs (in design) in advance
  - generality: over analytic/numerical techniques
  - detail: can simulate system details at arbitrary level
- drawbacks:
  - caution: does model reflect reality?
  - large scale systems: lots of resources to simulate (especially accurately simulate)
  - may be slow (computationally expensive – 1 min real time could be hours of simulated time)
  - art: determining right level of model complexity
  - statistical uncertainty in results

Classes of Network Simulation

Law and Kelton classify network simulations into the following classes:

- Static simulation - simulation using representation of a system at a particular time
- Dynamic simulation - simulation using representation of a system as it evolves over time
- Deterministic simulation - simulation using a model that does not contain any probabilistic (i.e. random) components
- Stochastic simulation - simulation using a model that contain probabilistic (i.e., random) components
- Continuous simulation - simulation involving an infinite number of events
- Discrete (event) simulation - simulation using finite number of events that take place during simulation at discrete times

Most network simulations are of dynamic, stochastic, and discrete nature

Network Simulation Packages/Tools

- Simulation-based performance evaluation can be performed using network simulation packages/tools
  - Open source (free!) tools:
    - Network Simulator v2 (NS2), REalistic And Large (REAL), Yet Another Tiny Simulator (YATS), and Global Mobile Simulator (GloMoSim), NCTU*ns*
  - Commercial tools:
    - OPNET
    - PacketStorm (emulator)
- More tools at: http://www.icir.org/models/simulators.html

Comparison of techniques

	Analytical Modeling	**Simulation**	**Measurement**
Accuracy	Moderate	Moderate	Various
Cost	Small	Medium	High
Time required	Small	Medium	High

Steps in Performance Evaluation

1. State objectives, define the system under study
2. List of system services and outcomes
3. Define and select performance metrics
4. Identify system & workload parameters that affect performance
5. Define factors to be studied (variation, level)
6. Select performance evaluation technique to be used
7. Select system workload i.e. list service requests to the system
8. Design the experiment(s)
9. Analyze and interpret data
   - stochastic processes, variability of data
10. Present results

Network Performance Metrics

The criterion used to quantify the performance of the system

Common metrics:

- Bandwidth
- Delay
- Loss
- Jitter
- Throughput
- Goodput
- Packet loss
- Utilization
- Queue length
- etc…

Derived metrics:

- TCP-friendliness
- Inter-protocol fairness
- Intra-protocol fairness
- Loss ratio
- Efficiency
- Packet retranmission ratio
- Layer subscription
- Mean queue length
- etc …

Learning Activity :

Examine the following network performance evaluation –based papers:

using Measurements method

- Fraleigh, C.; Moon, S.; Lyles, B.; Cotton, C.; Khan, M.; Moll, D.; Rockell, R.; Seely, T.; Diot, S.C., "**Packet-level traffic measurements from the Sprint IP backbone**," *IEEE Network,* vol.17, no.6 pp. 6- 16, Nov.-Dec. 2003

using Analytical Modeling method

- Lo Piccolo, F.; Neglia, G., "**The effect of heterogeneous link capacities in BitTorrent-like file sharing systems**" Proceedings of the International Workshop on Hot Topics in Peer-to-Peer Systems, 2004. pp. 40- 47, 8 Oct. 2004

Give your comment(s) for each method used.

Summary

- Introduce and overview system/network performance evaluation
- Identify reasons for performing performance evaluation
- Describe three commonly used techniques for system/network performance evaluation
- Explain the drawbacks of these techniques
- Discuss the advantages of these techniques
- Compare the three evaluation techniques
- Steps in system/network performance evaluation
- Performance metrics

Programming a simulation*

What 's in a simulation program?

- *simulated time:* internal (to simulation program) variable that keeps track of simulated time
- *system "state":* variables maintained by simulation program define system "state"
  - e.g., may track number (possibly order) of packets in queue, current value of retransmission timer
- *events:* points in time when system changes state
  - each event has associate *event time*
    - e.g., arrival of packet to queue, departure from queue
    - precisely at these points in time that simulation must take action (change state and may cause new future events)
  - model for time between events (probabilistic) caused by external environment

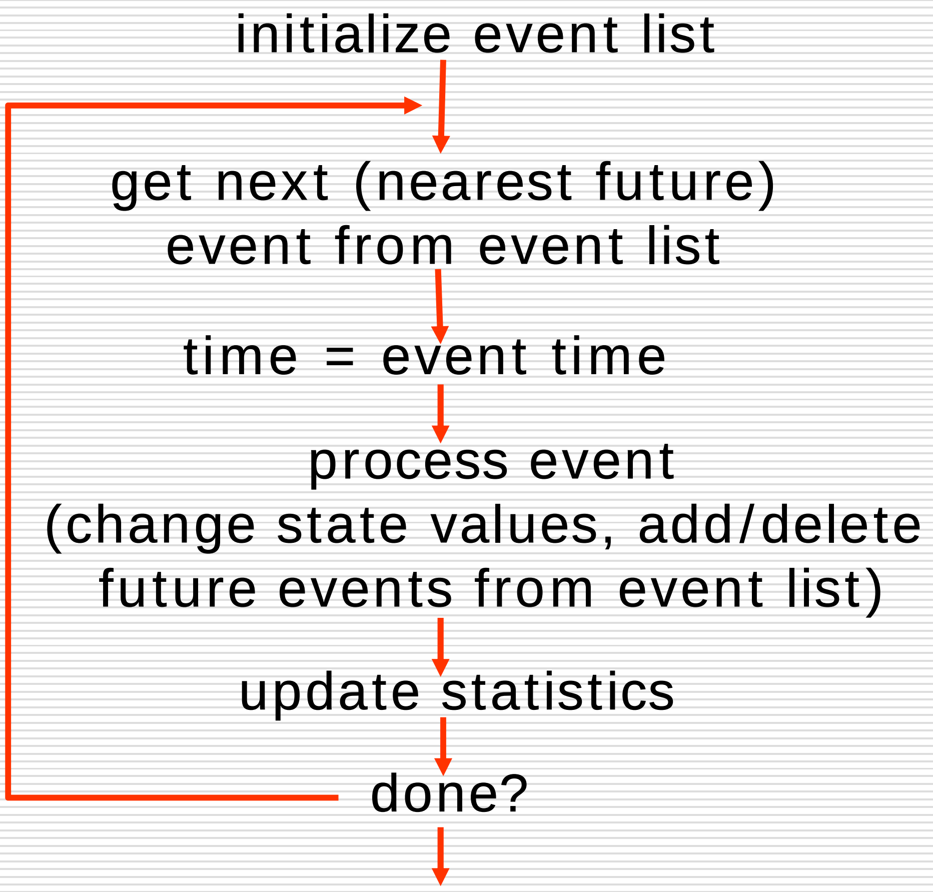
Simulator Structure*

- simulation program maintains and updates list of future events: event list

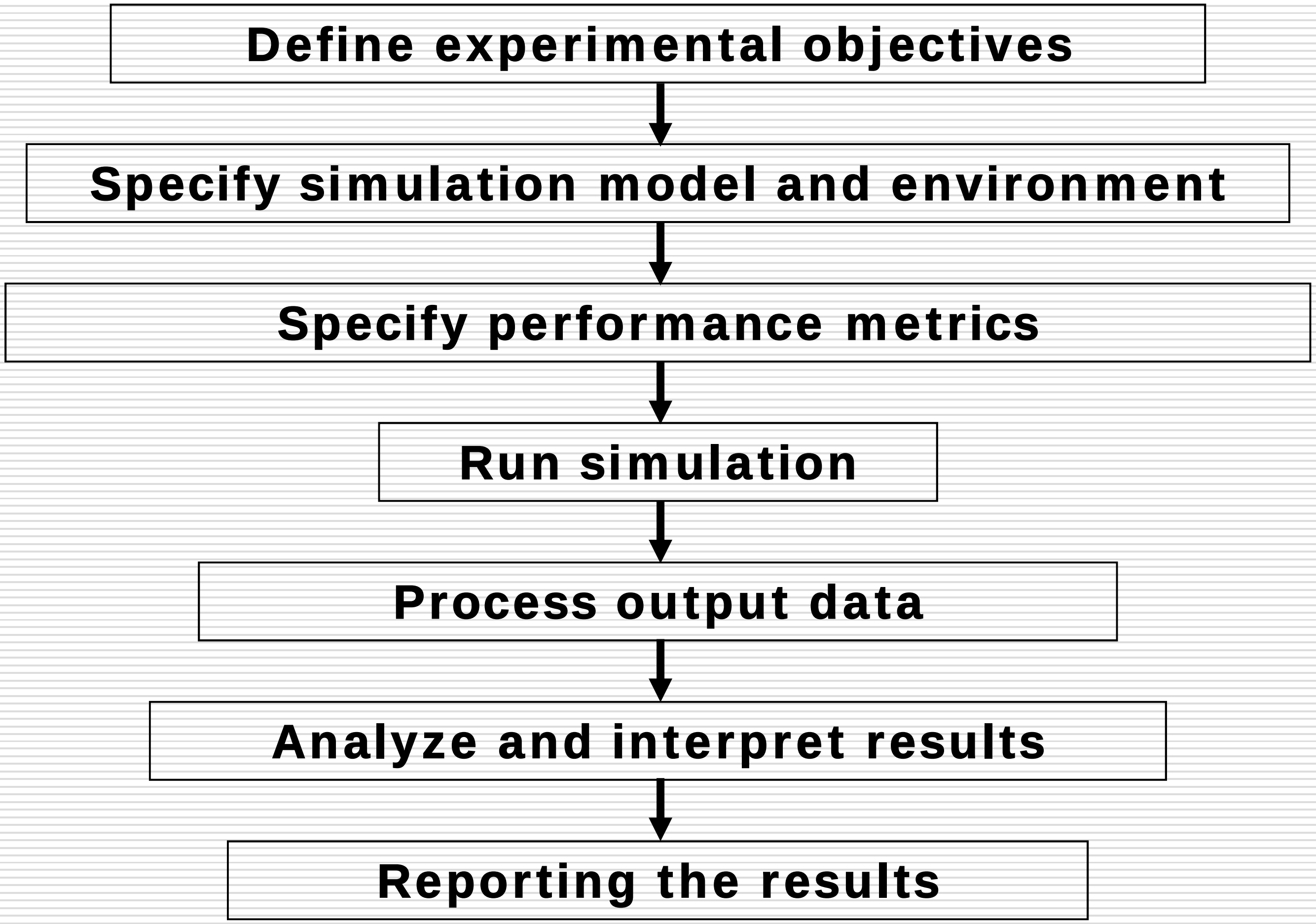
Need:

- well defined set of events
- for each event: simulated system action, updating of event list

Simulator Block Diagram*


initialize event list
get next (nearest future) event from event list
time = event time
process event (change state values, add/delete future events from event list)
update statistics
done?

Simulation Construction


Define experimental objectives
Specify simulation model and environment
Specify performance metrics
Run simulation
Process output data
Analyze and interpret results
Reporting the results

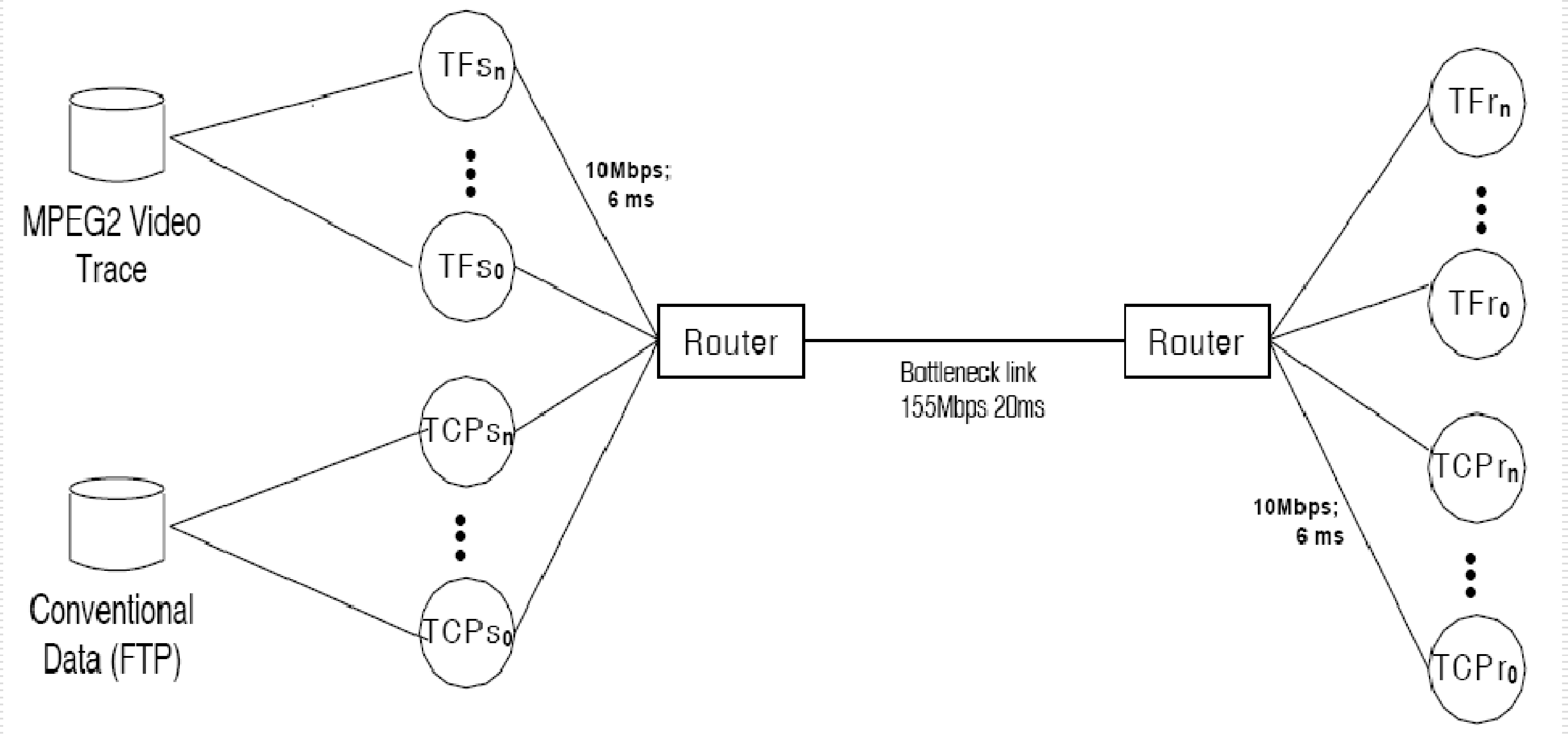
Define Experimental Objectives

- map research objectives into a set of simulation experiments
- state goals, objectives and evaluation criteria (metrics)

Define Simulation Models & Environment

- analyze the problems to be tackled
- specify a simulation model
- describe:
  - simulation scenario
  - topology
  - network environment
  - traffic characteristics
  - simulation parameters (including parameter sensitivity)

Example: Simulation Model / Topology



Simulation Metrics

The criterion used to quantify the performance of the system

Common metrics:

- Bandwidth
- Delay
- Loss
- Jitter
- Throughput
- Goodput
- Packet loss
- Utilization
- Queue length
- etc…

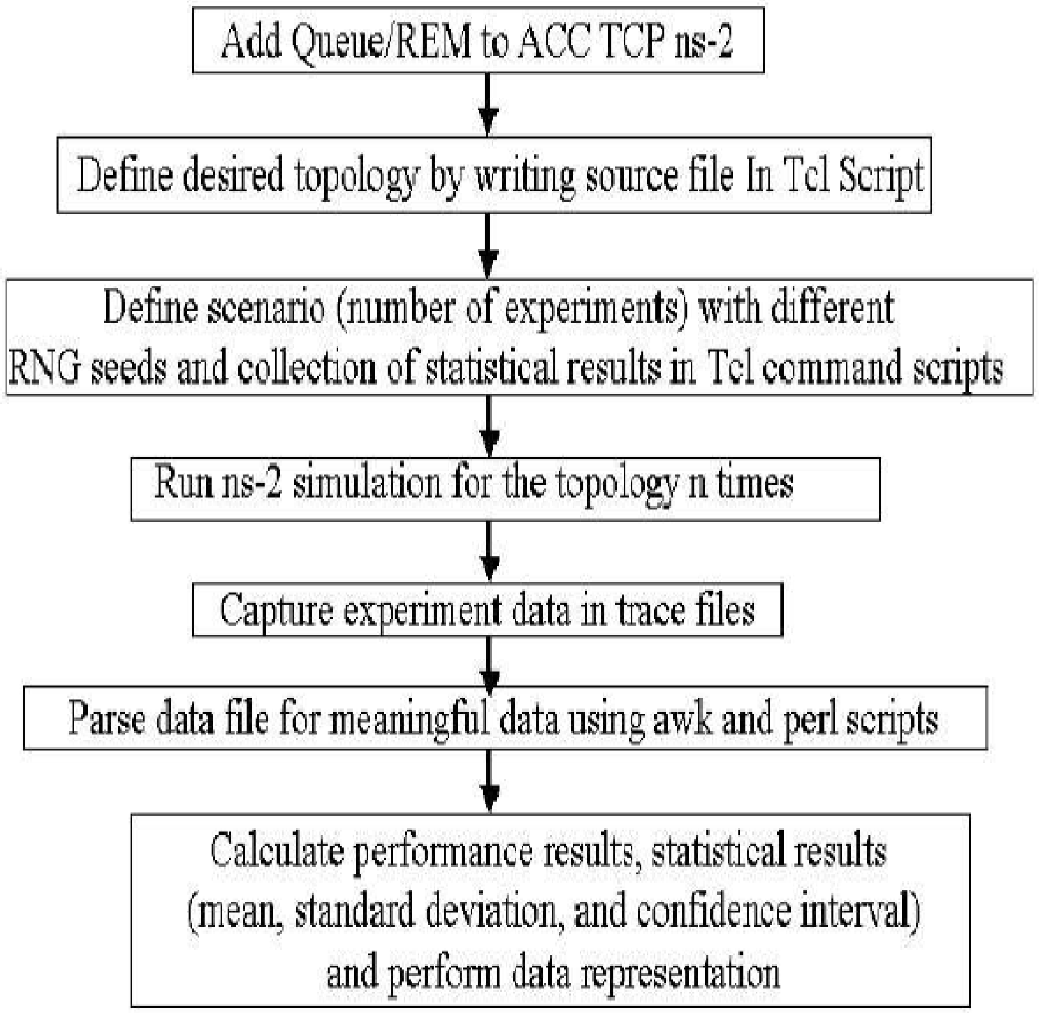
Derived metrics:

- TCP-friendliness
- Inter-protocol fairness
- Intra-protocol fairness
- Loss ratio
- Efficiency
- Packet retransmission ratio
- Layer subscription
- Mean queue length
- etc …

Simulation Phase

- put together simulation model, performance metrics and environment (scenarios, topology and configuration) using network simulator input scripts and run.
- Example:
  - run simulation 20 times each using different *Random Number Generator (RNG)* seeds
  - get the simulation results, use appropriate statistical function (if needed)
  - quote result with error bars with respect to confidence intervals of 95%

NS2 Experiment Example


Add Queue/REM to ACC TCP ns-2
Define desired topology by writing source file In Tcl Script
Define scenario (number of experiments) with different RNG seeds and collection of statistical results in Tcl command scripts
Run ns-2 simulation for the topology n times
Capture experiment data in trace files
Parse data file for meaningful data using awk and perl scripts
Calculate performance results, statistical results (mean, standard deviation, and confidence interval) and perform data representation

Simulation Validation & Verification

- Process of assuring that a simulation model provides meaningful answers to the questions being investigated
- Process to verify that the simulation model provide enough representation of a real system
- Pawlikowski et al. revealed a crisis of mistaken analyses of network simulation results:
  - not concerned about the random nature of output data obtained from stochastic simulation studies
  - did not care to mention that final results were outcomes of appropriate statistical analyses
  - reported purely random results

Validation & Verification (cont…)

- Issues:
  - number of simulation repetitions?
    - using different RNG seeds?
    - replication method
      - conduct simulations by ***n*** independent runs
    - batch means
      - one long run is performed, ***n*** performance measures are conducted
  - Confidence interval?
  - how long simulation is run?
    - avoiding warm-up period

Post-Simulation Phase

- Properly deal with captured data using appropriate tools
- Be aware of these important tools:
  - scripting tool for parsing data (e.g. perl, awk)
  - statistical tools for complex statistical analysis (e.g.: spss, minitab, sas)
  - gnuplot (vs. ms-excel) for data presentation
  - document processing tools, e.g. LaTeX or LyX (vs. word processor e.g. ms-word)
  - bibliographical tools (e.g. Bibtex (gbib, bibedit) vs endnote)
  - Other Linux (Unix) tools such as (Gimp, postscript tools, etc)

Introduction to Network Simulation: A NS2 Tutorial

Evaluating Network Performance Using Simulation Technique

Objectives of this tutorial

- Introduce one of the most useful tools in networking research and development.
- Understand and work with a popular network simulator.
- Get a better understanding of the networking dynamics.

The NS2 Simulator

- NS2 stands for Network Simulator version 2.
- NS2:
  - Is a discrete event simulator for networking research.
  - Works at packet level.
  - Provides support to simulate protocols such as TCP, UDP, FTP, HTTP and DSR.
  - Simulate wired and wireless network.
  - Is primarily Unix based.
  - Use TCL as its scripting language.
- NS2 is a standard experiment environment in research community.

The NS2 Simulator

- a discrete event simulator
  - models a system as it evolves over time
  - state variables change instantaneously at separate point in time
- began as variant of the REAL network simulator in 1989
- targeted at networking research
- provides substantial support for simulation of:
  - transport layer protocols – e.g. TCP, UDP, TFRC etc
  - network layer protocols, routing, IPv6, mobile IP etc
  - multicast protocols over wired and wireless (local and satellite) networks

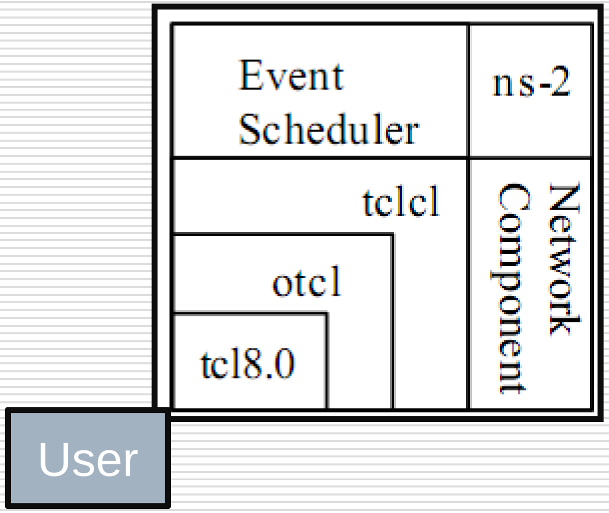
- More at the NS2 site: http://www.isi.edu/nsnam/ns/

The NS2 Simulator

- Contain package of tools that simulates behavior of networks
  - Create Network Topologies
  - Log events that happen under any load
  - Analyze events to understand the network behavior

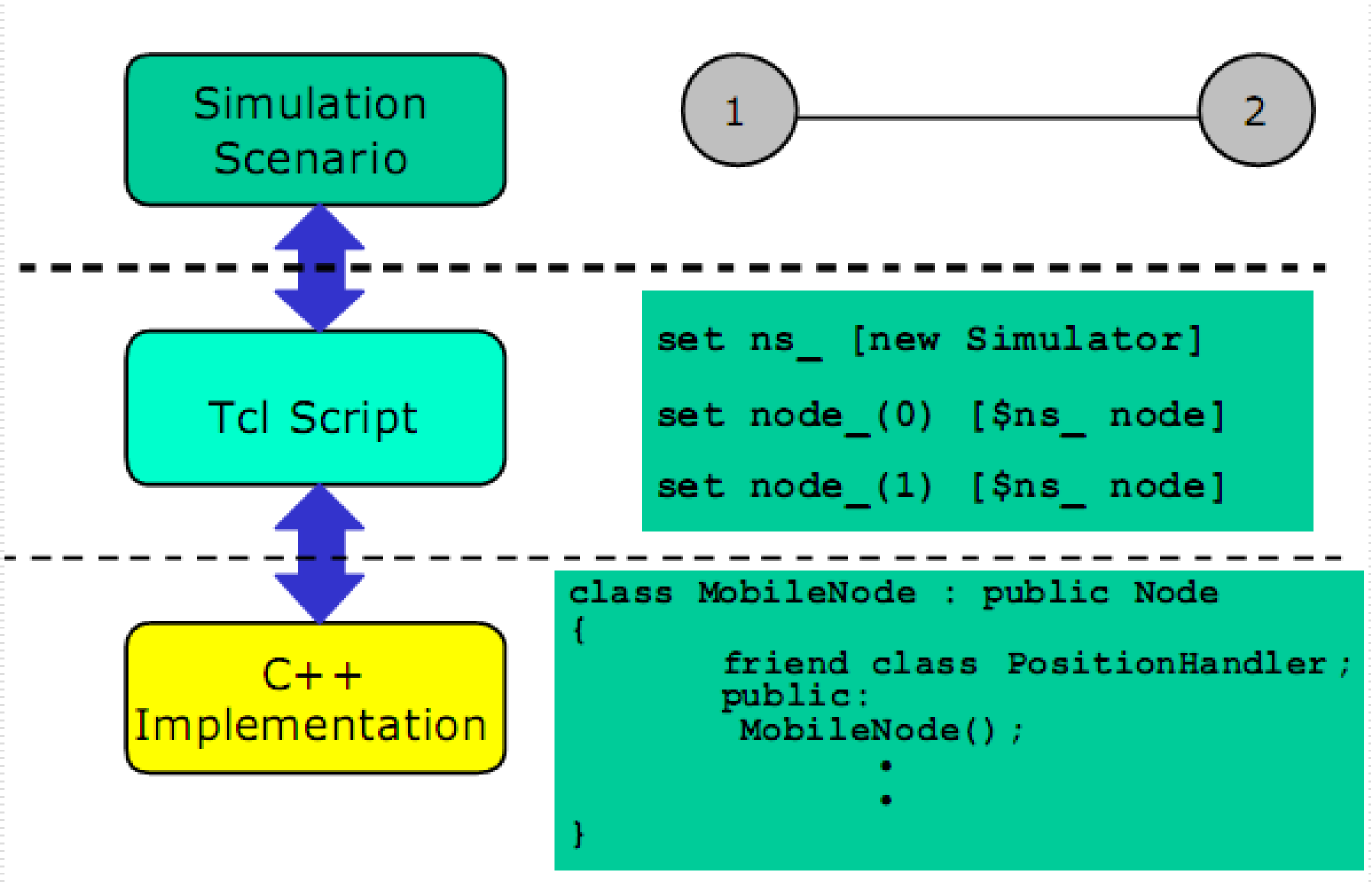
- It is an open source
  - – available at source forge
  - – http://www.isi.edu/nsnam/ns/

What is NS2 (cont.)?



- otcl: Object-oriented support
- tclcl: C++ and otcl linkage
- Discrete event scheduler
- Data network (the Internet) components.

NS2 implementation



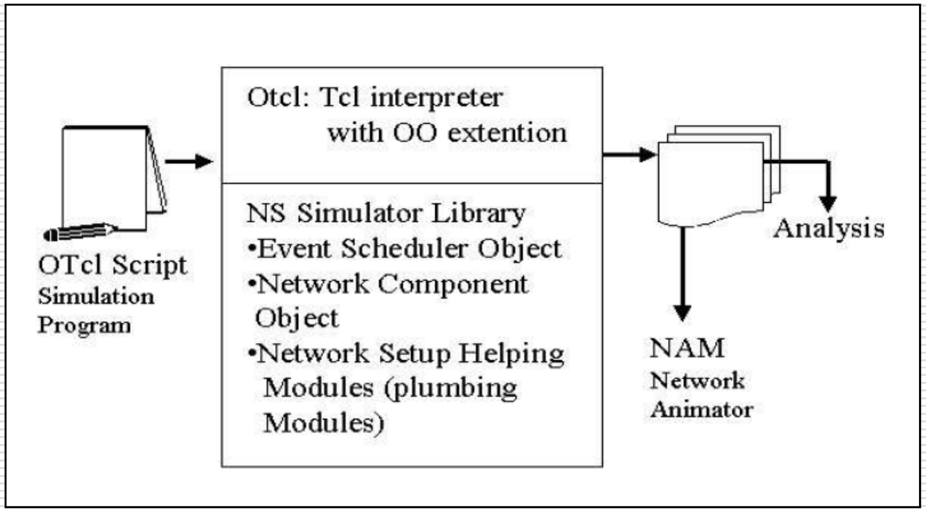
```
set ns_ [new Simulator]

set node_(0) [$ns_ node]

set node_(1) [$ns_ node]
```

```
class MobileNode : public Node
{
        friend class PositionHandler;
        public:
         MobileNode();
                   .
                   .
}
```

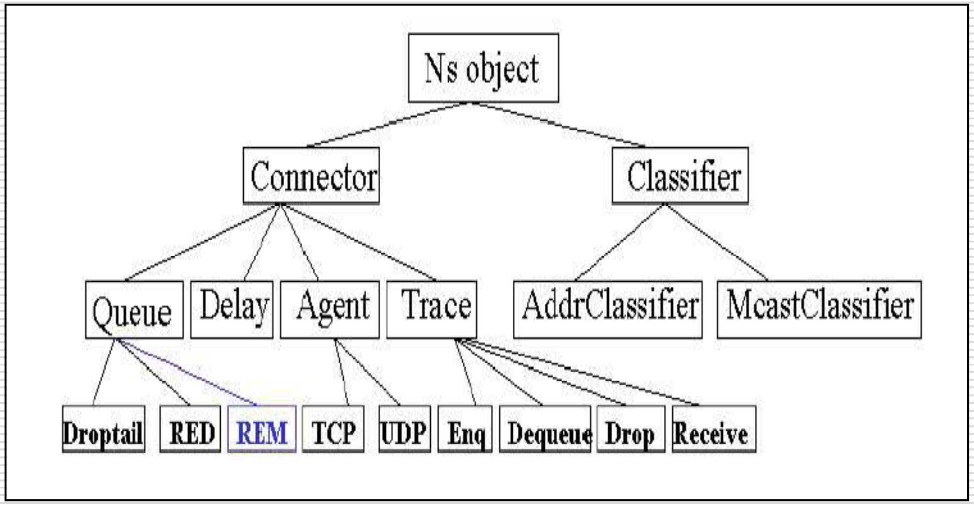
Why Tcl & C++ ?

- C++: Detailed protocol simulations require systems programming language
  - byte manipulation, packet processing, algorithm implementation
  - Run time speed is important
  - Turn around time (run simulation, find bug, fix bug, recompile, re-run) is slower.
- Tcl: Simulation of slightly varying parameters or configurations
  - quickly exploring a number of scenarios
  - iteration time (change the model and re -run) is more important.

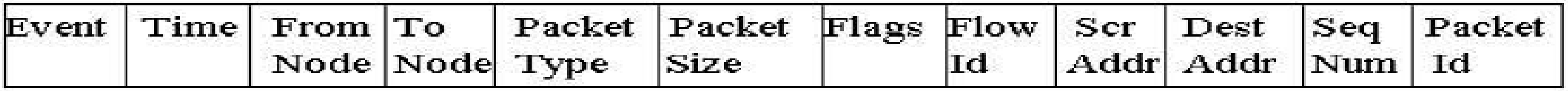
NS2 Simulation Environment


Otcl: Tcl interpreter with OO extention
NS Simulator Library
•Event Scheduler Object
•Network Component Object
•Network Setup Helping Modules (plumbing Modules)
OTcl Script Simulation Program
Analysis
NAM Network Animator

NS2 process and result generation



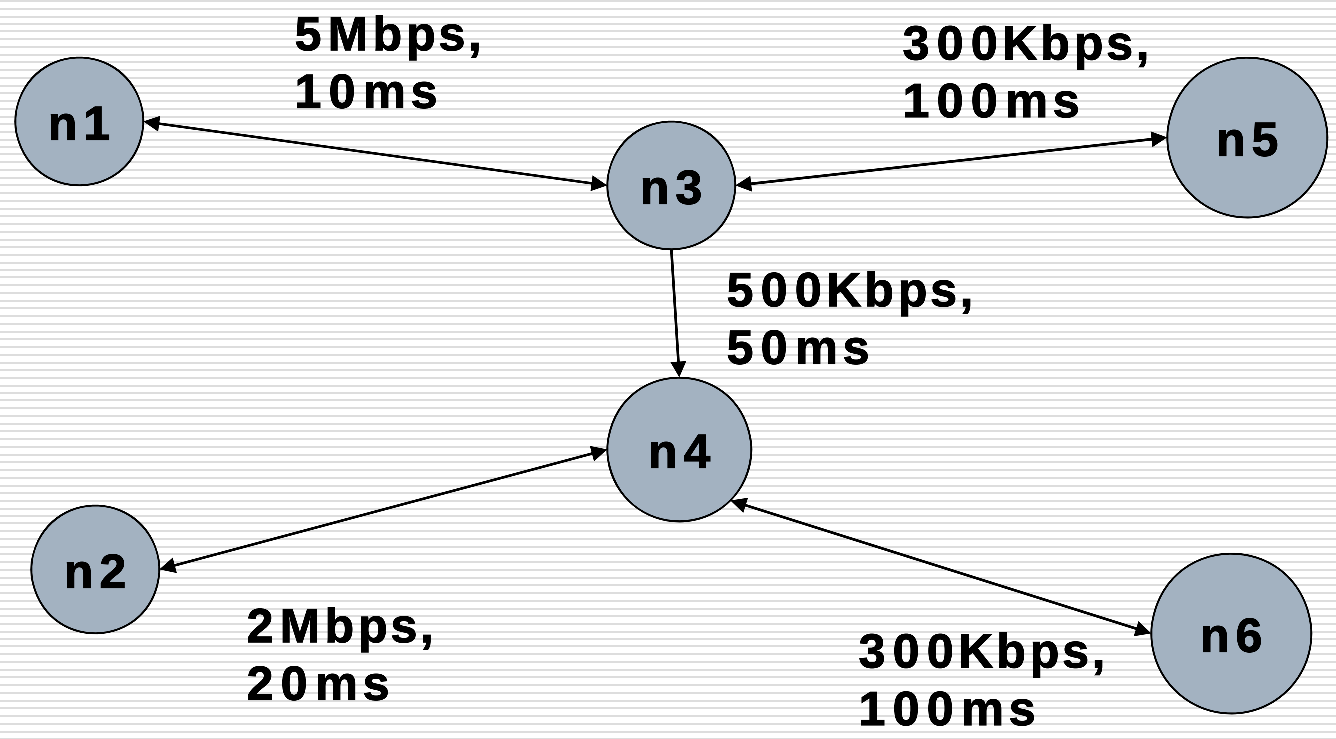
NS2 objects



Event	Time	From Node	To Node	Packet Type	Packet Size	Flags	Flow Id	Scr Addr	Dest Addr	Seq Num	Packet Id

NS2 trace structure

Creating Topologies

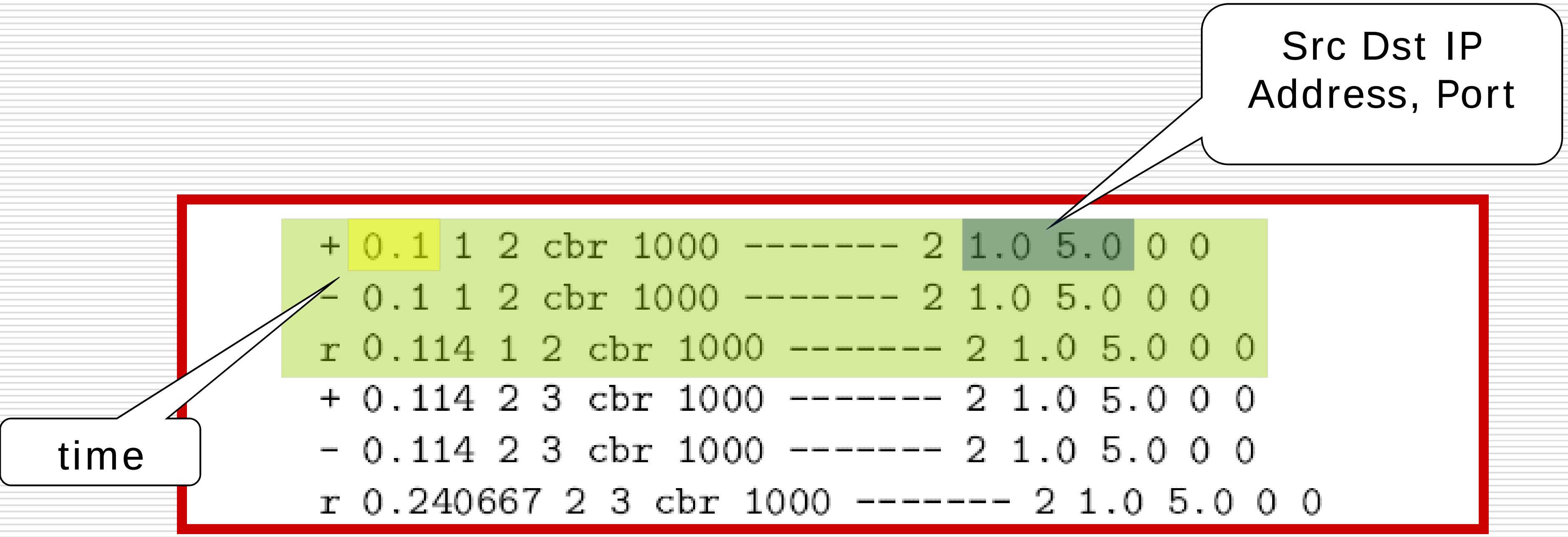


Creating Topologies

- Nodes
  - Set properties like queue length, location
  - Protocols, routing algorithms
- Links
  - Set types of link – Simplex, duplex, wireless, satellite
  - Set bandwidth, latency etc.
- Done through tcl Scripts

Observing Network Behavior

- Observe behavior by tracing “events”
  - Eg. packet received, packet drop etc.

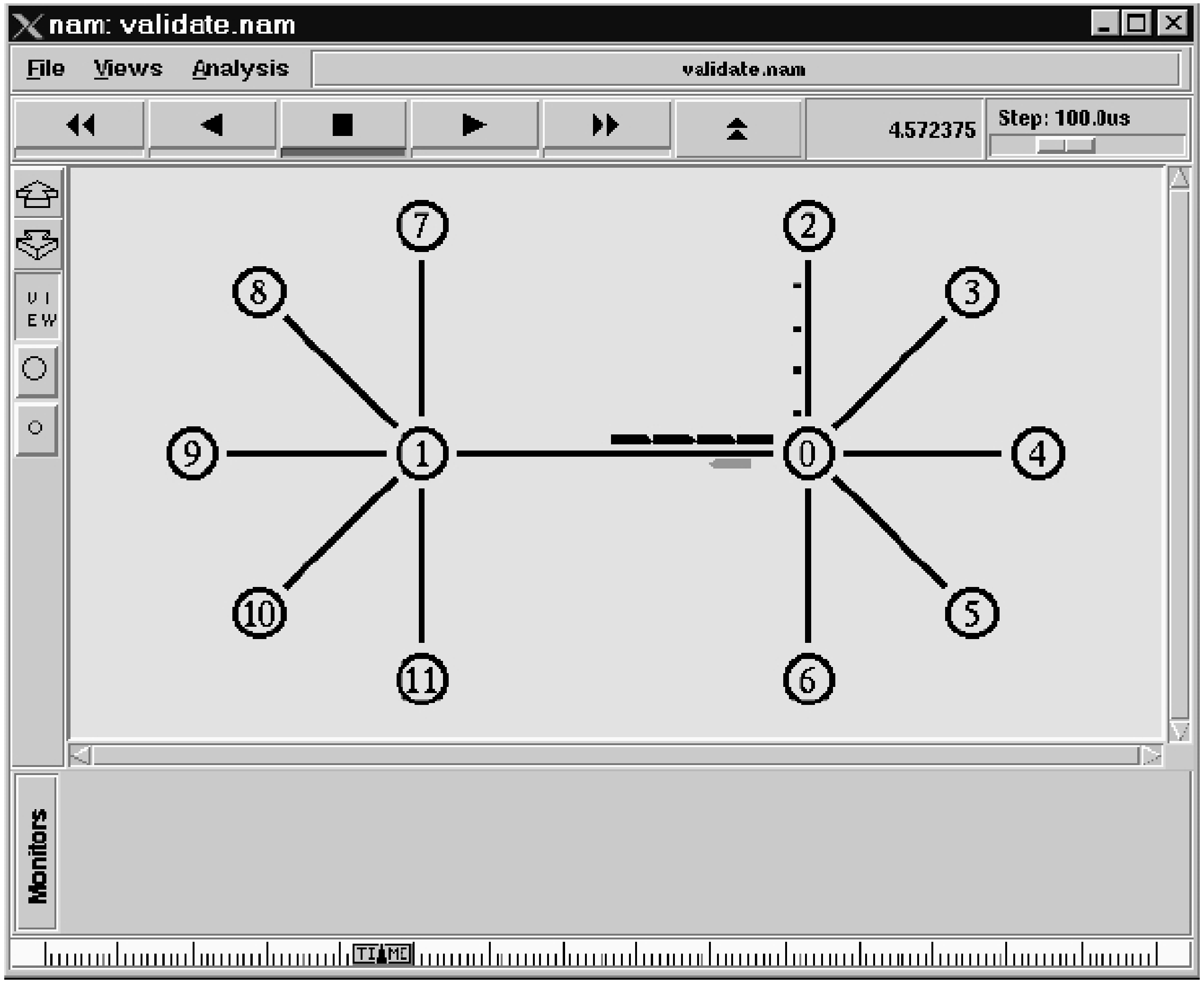


```
+ 0.1 1 2 cbr 1000 ------- 2 1.0 5.0 0 0
- 0.1 1 2 cbr 1000 ------- 2 1.0 5.0 0 0
r 0.114 1 2 cbr 1000 ------- 2 1.0 5.0 0 0
+ 0.114 2 3 cbr 1000 ------- 2 1.0 5.0 0 0
- 0.114 2 3 cbr 1000 ------- 2 1.0 5.0 0 0
r 0.240667 2 3 cbr 1000 ------- 2 1.0 5.0 0 0
```

Observing Network Behavior

- NAM:
  - Network Animator
  - A visual aid showing how packets flow along the network
- We’ll see a demo..

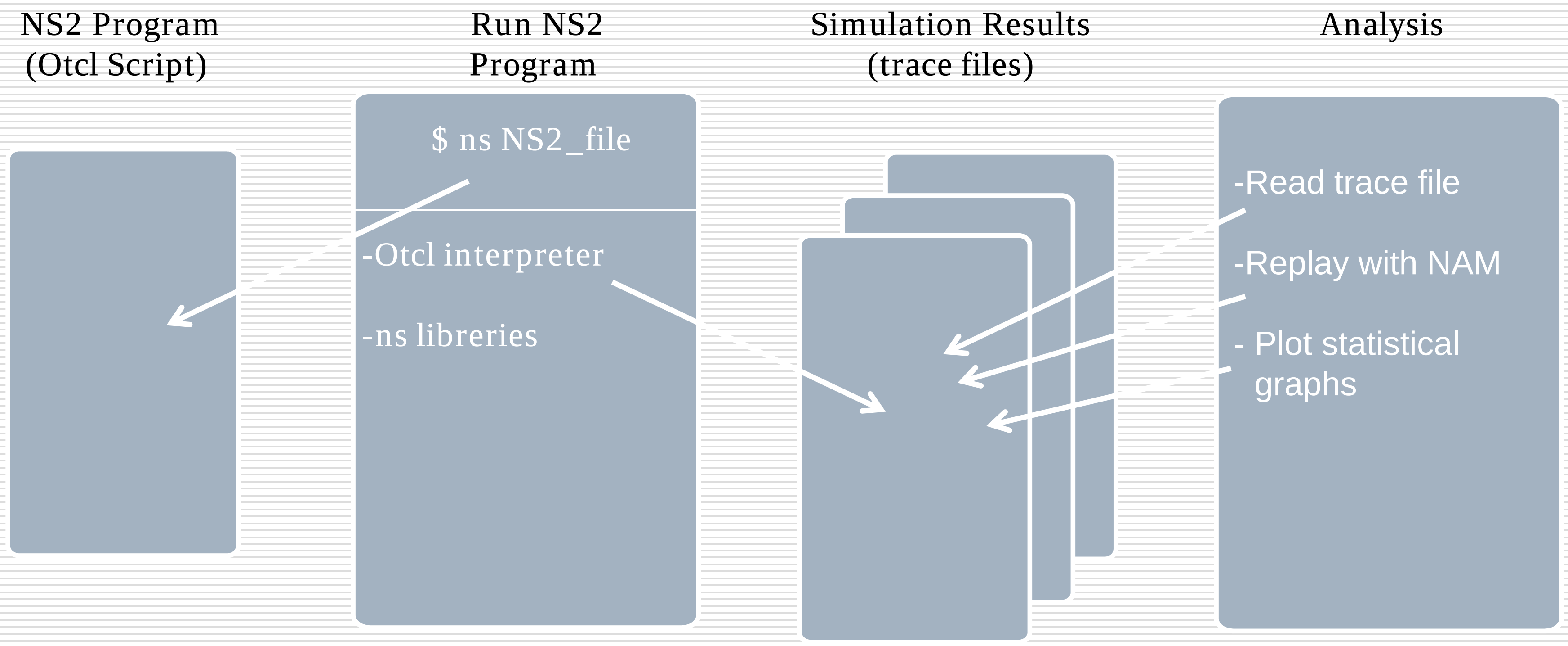
Simulation Visualization



… Now, How Do I use NS2?

- Creating a Simple Topology
- Getting Traces
- Using NAM

Running NS2 Program



Hello World

- simple.tcl:

```
set ns [new Simulator]
$ns at 1 “puts  \“Hello World!\””
$ns at 1.5 “exit”
$ns run
```

- [group@uum]$ ns simple.tcl
- Hello World!
- [group@uum]$

Basic tcl

```
proc test {} {
        set a 43                                  ; a = 43
        set b 27                                  ; b = 27
        set c [expr $a + $b]                      ; c = a + b
        set d [expr [expr $a - $b] * $c]          ; d = (a – b) * c
        for {set k 0} {$k < 10} {incr k} {        ; for (k=0; k<10; k++)
                puts “k = $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 a [new mom]
$a set age_ 45
          set b [new kid]
          $b set age_ 15

 $a greet
 $b greet
```

Basic NS2

- Create a new simulator object
- [Turn on tracing]
    - [Open your own trace files]
- Create network (physical layer)
- Create link and queue (data-link layer)
- Define routing protocol
- Create transport connection (transport layer)
- Create traffic (application layer)
- Insert errors

Basics of using NS2

- Define Network topology, load, output files in Tcl Script
- To run,

```
$ ns simple_network.tcl
```

- Internally, NS2 instantiates C++ classes based on the tcl scripts
- Output is in form of trace files

Creating simulator instance

- Create simulator instance
  - set ns [new Simulator]
    - Usually the first non-comment statement in ns-script
    - Initialize the packet format
    - Create a scheduler (default is a calendar scheduler)
    - Create a "null agent"

Turning on a trace file

- Open file for NS tracing

```
set f [open out.tr w]
$ns trace-all $f
```

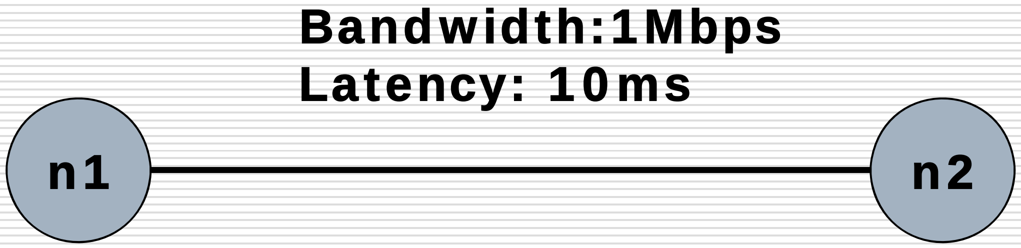
- Open file for nam tracing

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

- Open your own trace file

```
set my_f [open my_out.tr w]
puts $my_f  “[$ns now] [expr $x(1) + $y(1)]”
```

Easy Example – Creating the topology



Creating the topology

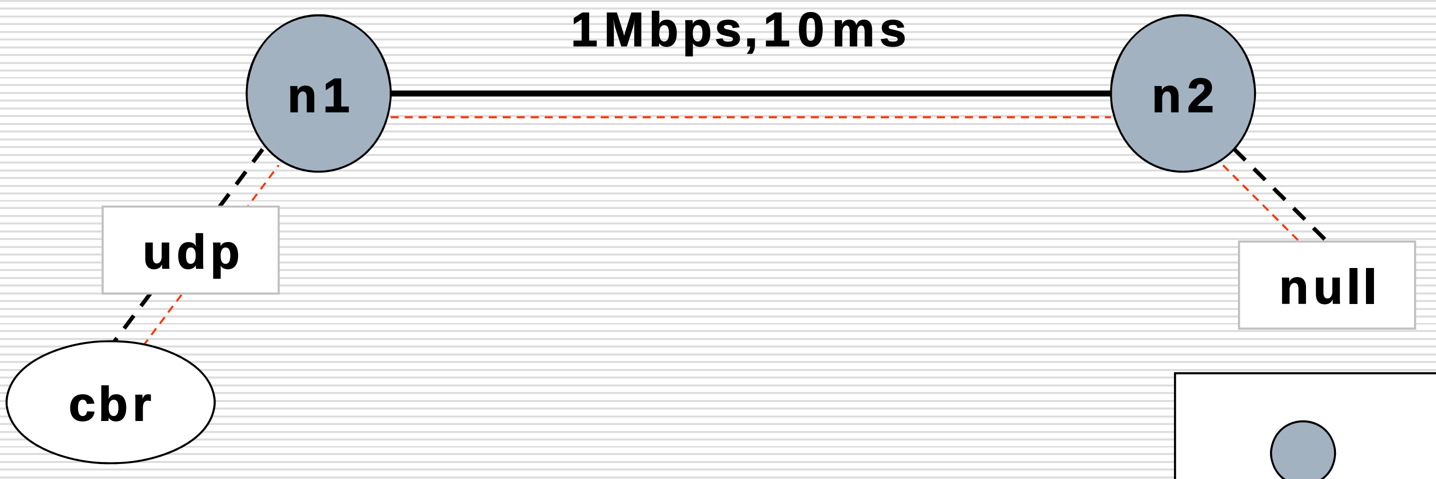
```
#create a new simulator object
set ns [new Simulator]

#open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace

    #close the trace file
    close $nf

    #execute nam on the trace file
    exec nam out.nam &

    exit 0
}
```

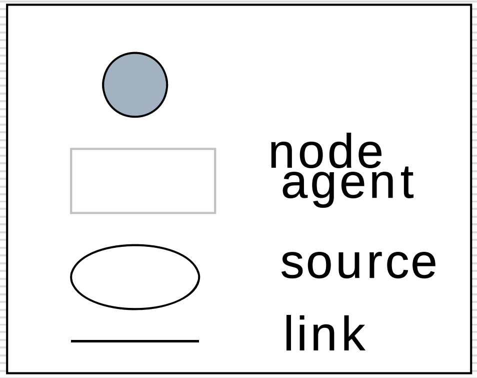
Creating the topology (Contd)

```
#create two nodes
set n0 [$ns node]
set n1 [$ns node]

#create a duplex link between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

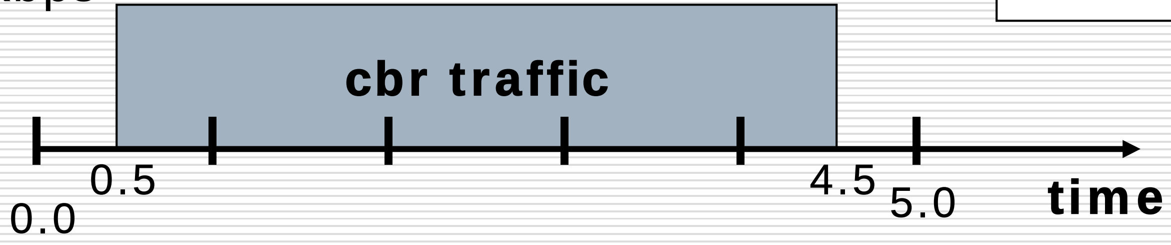
Demo

Adding traffic





Packet Size: 500 bytes
rate: 800Kbps

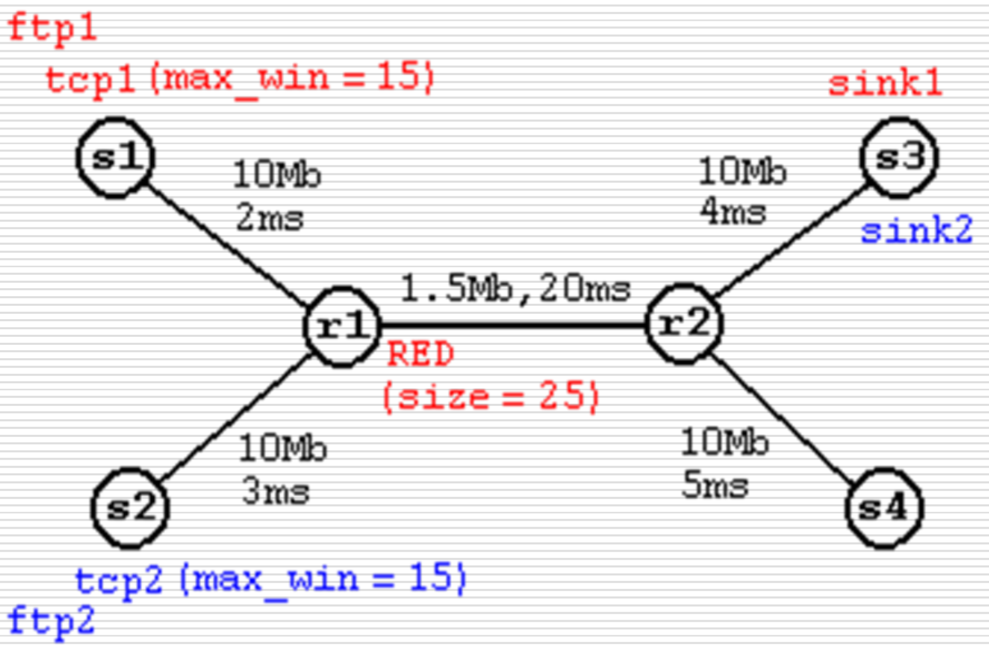


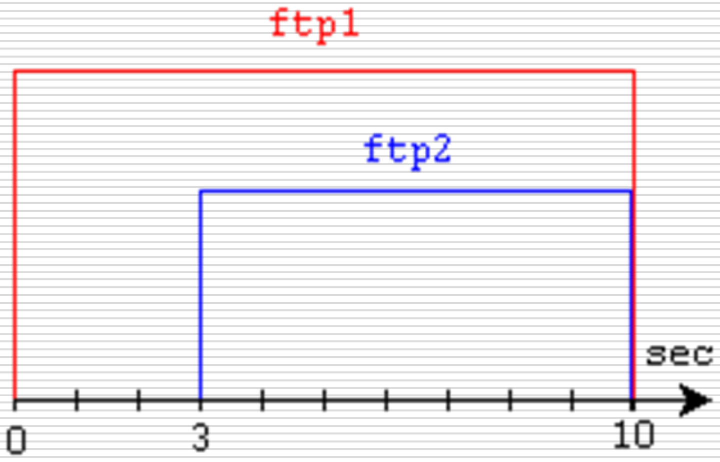
Putting it together..

```
#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

#create a Null agent(a traffic sink) and attach it to node n1
set null0 [new Agent/Null]
$ns attach-agent $n1 $null0

#Connect the traffic source to the sink
$ns connect $udp0 $null0

#Schedule events for CBR traffic
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

#call the finish procedure after 5 secs of simulated time
$ns at 5.0 "finish"

#run the simulation
$ns run
```

Another Example: Creating a Topology

- Network topology





Another Example: Cont …

- Creating nodes

```
set node_(s1) [$ns node]
set node_(s2) [$ns node]
set node_(r1) [$ns node]
set node_(r2) [$ns node]
set node_(s3) [$ns node]
set node_(s4) [$ns node]
```

Another Example: Cont …

- Creating Link and Queue

```
$ns duplex-link $node_(s1) $node_(r1) 10Mb 2ms DropTail
$ns duplex-link $node_(s2) $node_(r2) 10Mb 3ms DropTail
$ns duplex-link $node_(r1) $node_(r2) 1.5Mb 20ms RED
$ns queue-limit $node_(r1) $node_(r2) 25
……
```

Creating a TCP connection

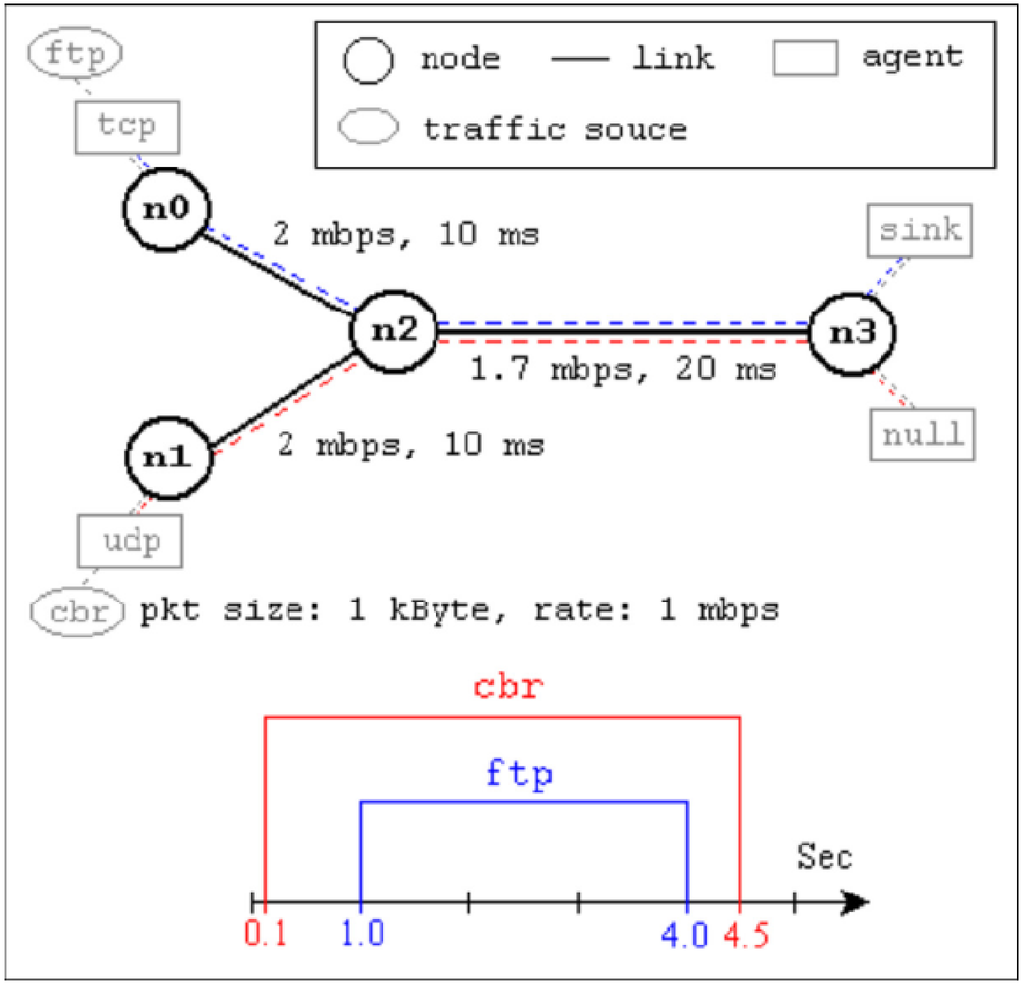
- set tcp [new Agent/TCP/Reno]
- $ns attach-agent $s1 $tcp
- set sink [new Agent/TCPSink]
- $ns attach-agent $s4 $sink

Creating traffic

- Attaching FTP traffic on the top of TCP

```
set ftp [new application/FTP]
$ftp attach-agent $tcp
```

3rd Scenario* (from NS by Example)

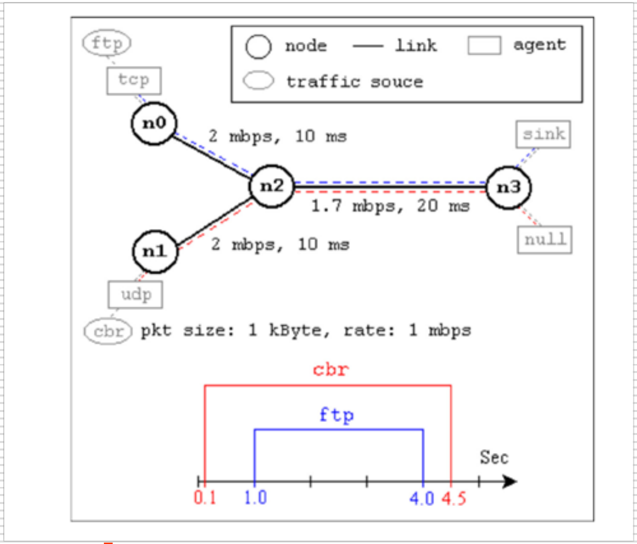


NS by Example by Jae Chung **and** Mark Claypool

3rd Scenario* (from NS by Example)

```
#Create a simulator object
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}
```

3rd Scenario* (from NS by Example)



```
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10
```

3rd Scenario* (from NS by Example)

```
#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5
```

3rd Scenario* (from NS by Example)

```
#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
```

To create agents or traffic sources, we need to know the class names these objects (Agent/TCP, Agent/TCPSink, Application/FTP and so on). This information can be found in the NS documentation. But one shortcut is to look at the "NS2/tcl/libs/ns-default.tcl" file.

```
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

3rd Scenario* (from NS by Example)

```
#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false
```

3rd Scenario* (from NS by Example)

```
#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run
```

Start / Stop ns

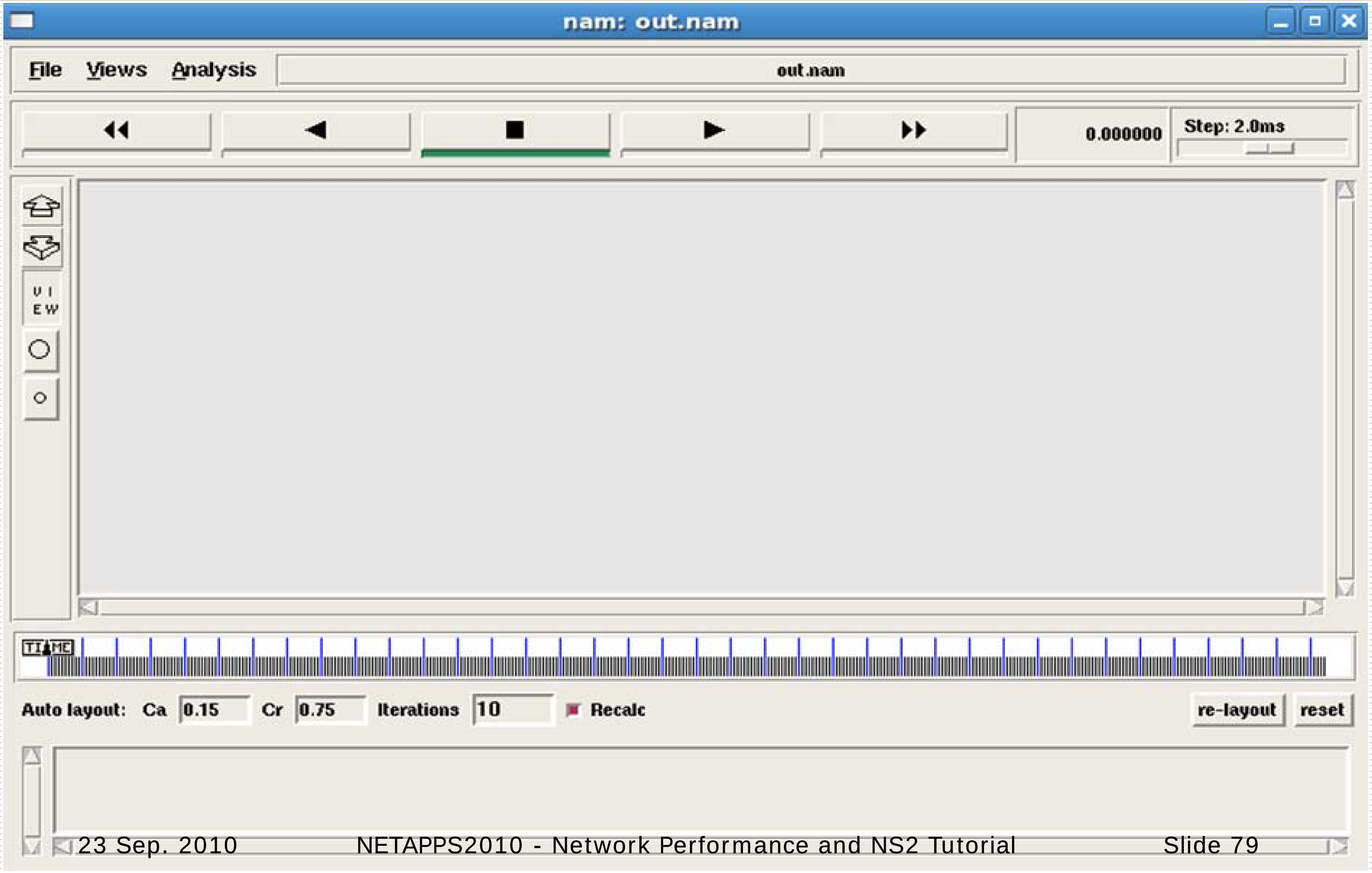
- Schedule an event to start traffic at time 3.0
  - $ns at 3.0 "$ftp start“
- Schedule an event to stop ns at time 10.0
  - $ns at 10.0 "$ftp stop“
- Start ns
  - $ns run
    - last statement
- Stop ns
  - exit 0

Visualization tool: nam

- Replay events from a nam trace file
- The nam trace file can be huge when simulation time is long or events happen intensively. Be careful!
- Run nam:
  - – $nam –a nam_trace_file.nam
  - – In NS2 script:

```
Proc finish{} {
……
exec nam –a nam_trace_file.nam &
Exit
}
```

NAM



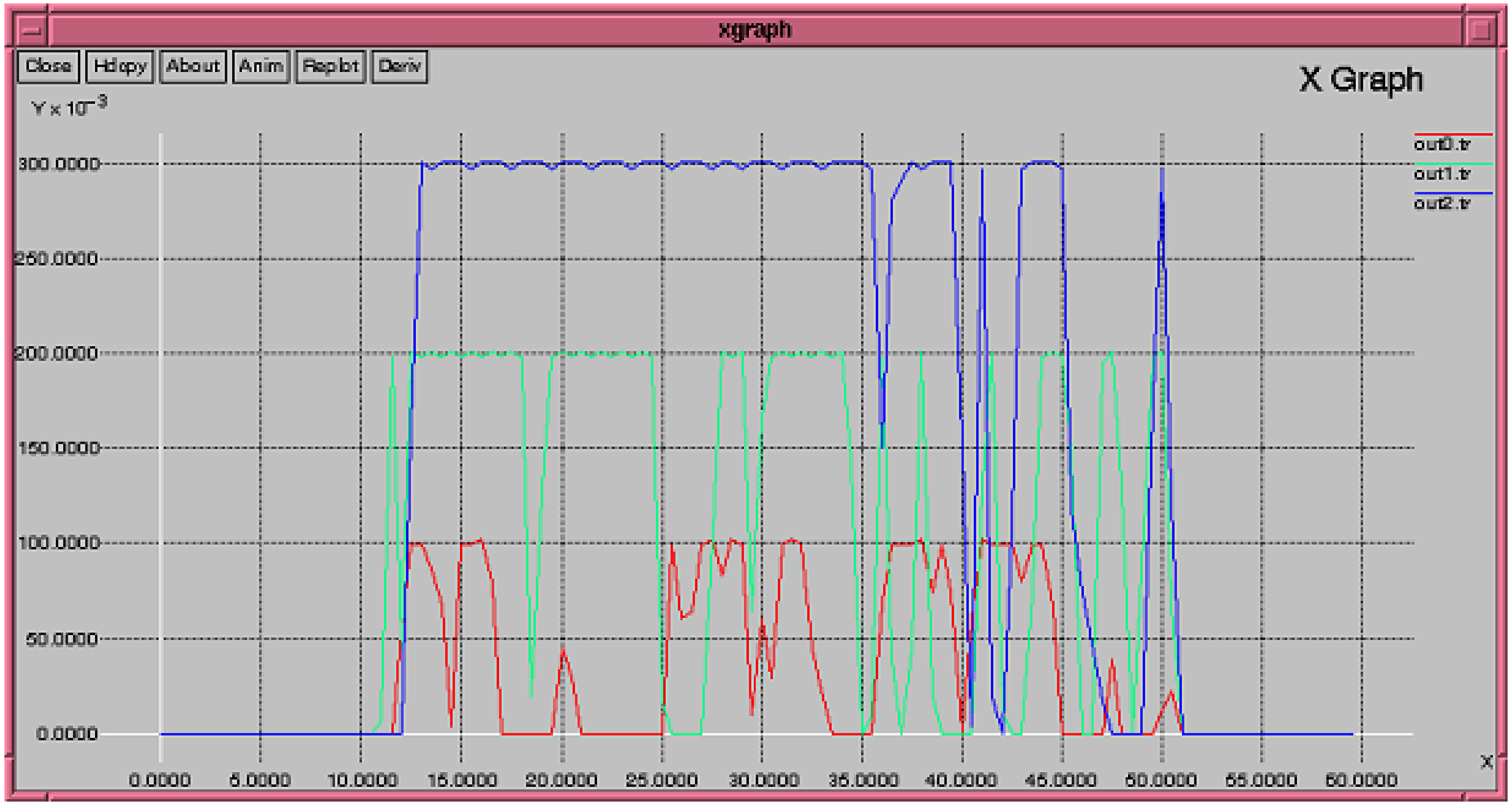
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc re-layout reset

Draw plots using xgraph

- Create your own output files
- Collect statistical data synchronized.
- Run xgraph:
    - – $xgraph out0.tr, out1.tr –geometry 800x400
    - – In NS2 script:

```
Proc finish{} {
……
exec xgraph out0.tr, out1.tr out2.tr –geometry  800x400 &
exit
}
```

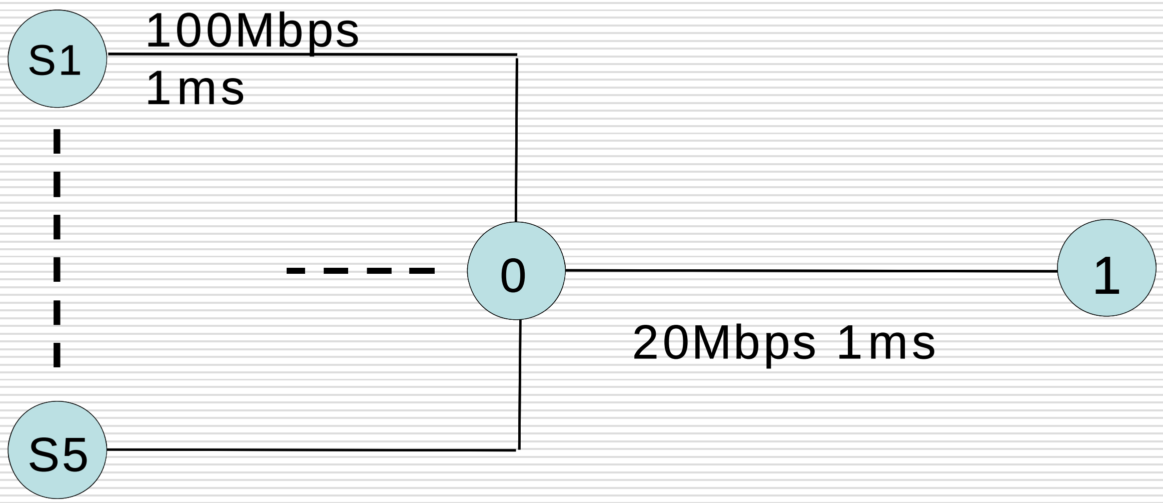
XGraph



xgraph
Close
Hdcpy
About
Anim
Replot
Deriv
X Graph
Y x 10^-3
300.0000
250.0000
200.0000
150.0000
100.0000
50.0000
0.0000
0.0000
5.0000
10.0000
15.0000
20.0000
25.0000
30.0000
35.0000
40.0000
45.0000
50.0000
55.0000
60.0000
X
out0.tr
out1.tr
out2.tr

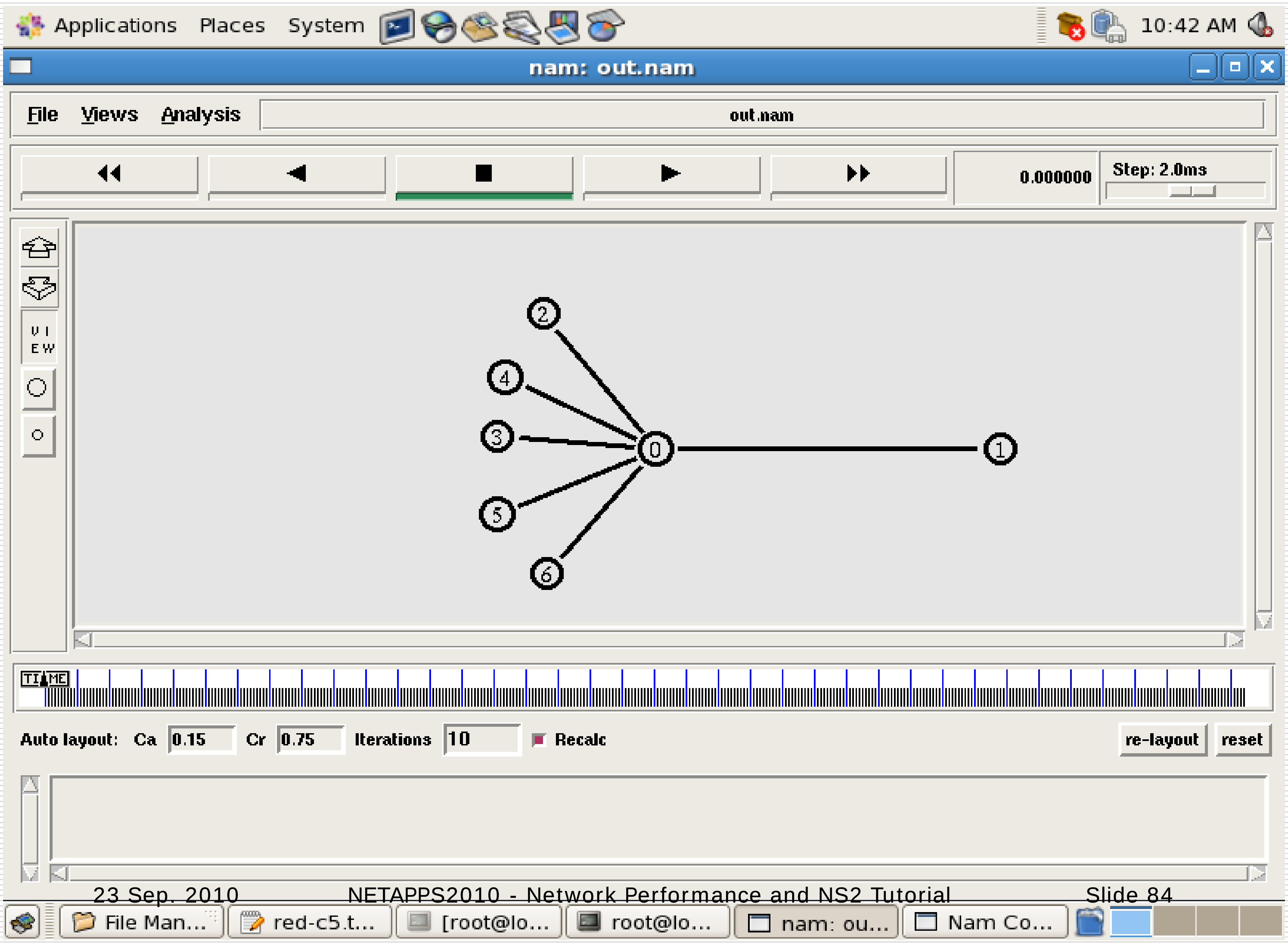
FN Queue Monitor Example

- Sets up the network topology and runs the simulation scenario shown in figure below.
- RED queue that can hold up to 50 packets is used for the link 0 - 1.
- See how the FN queue works by measuring the dynamics of current and average queue size.

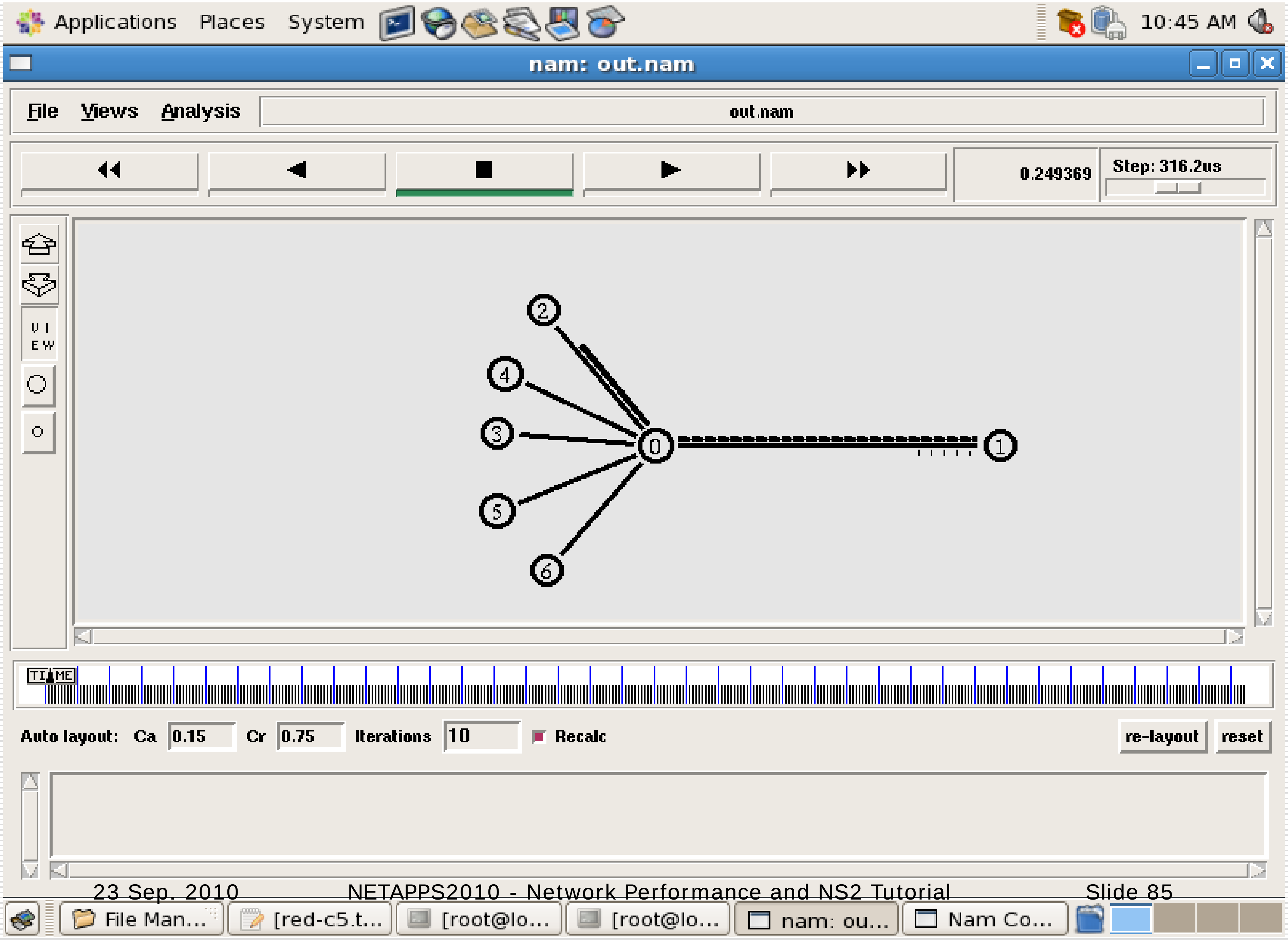


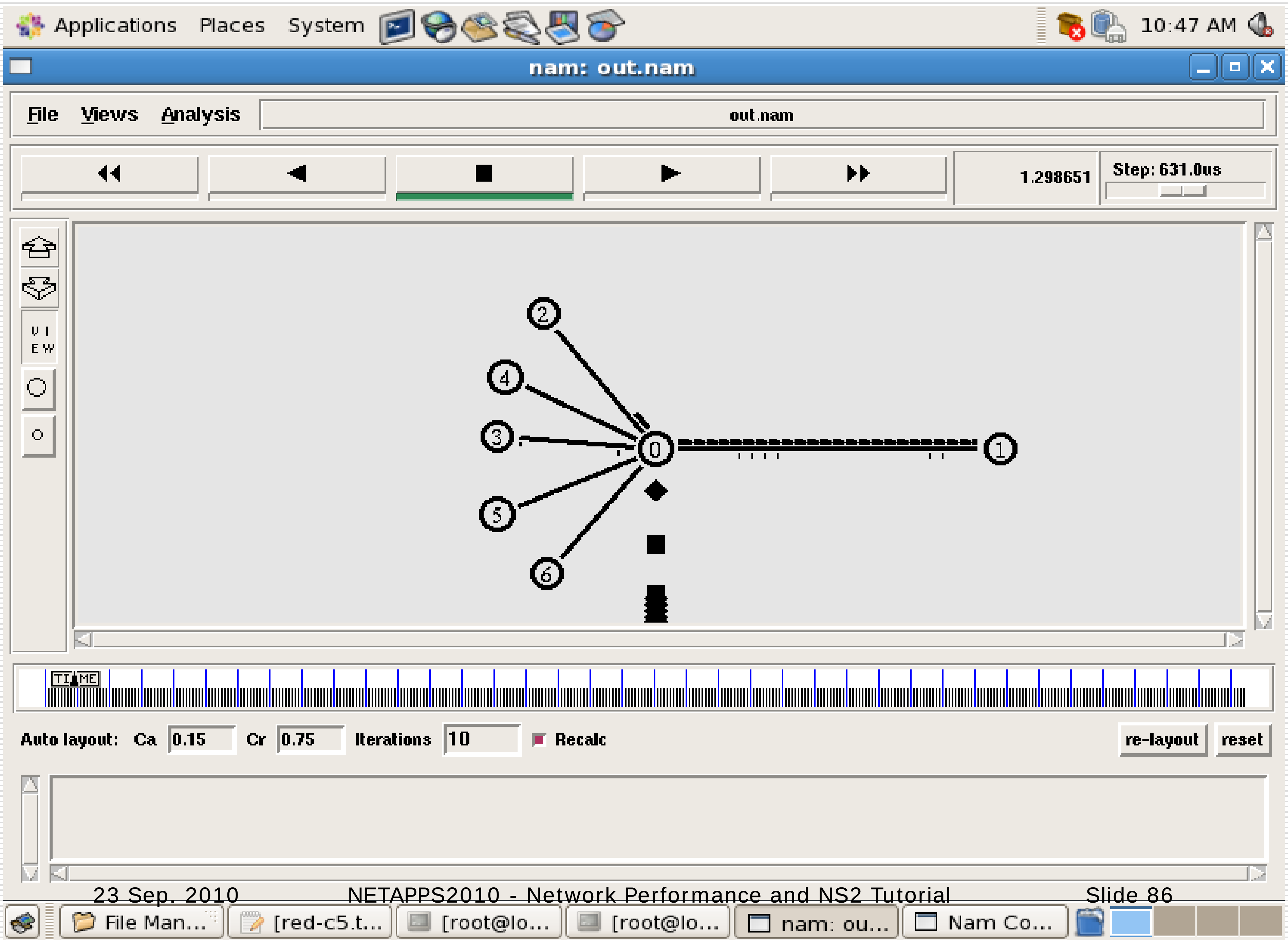
Tracing the queue

- #Queue monitoring
- set redq [[$ns link $n0 $n1] queue]
- set traceq [open fn-queue.tr w]
- $redq trace curq_
- $redq trace ave_
- $redq attach $traceq



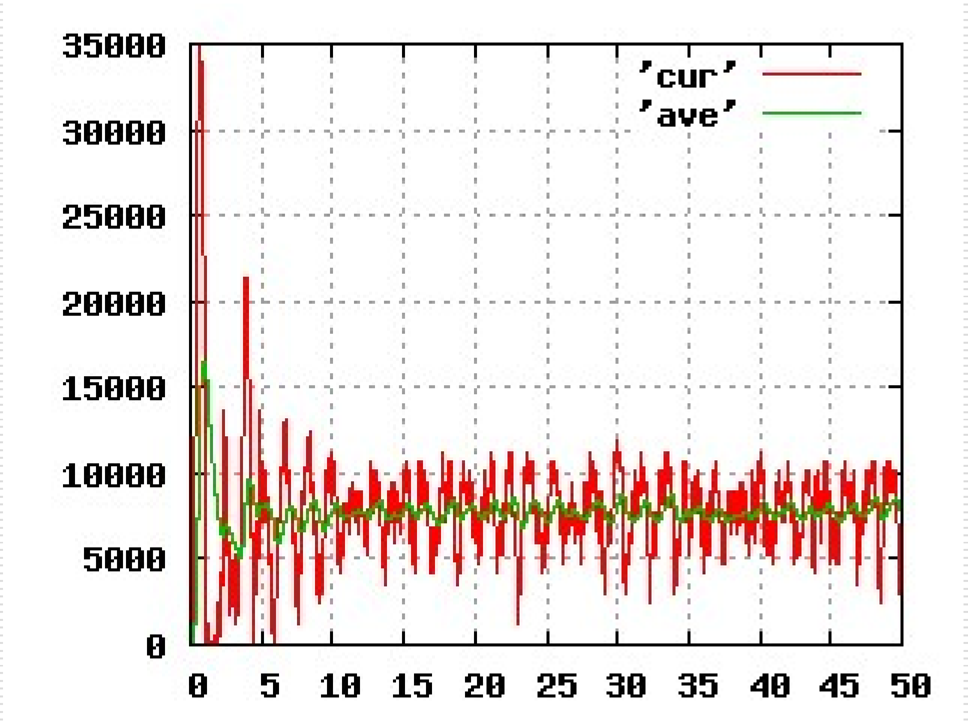
Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc


Applications
Places
System
10:45 AM
nam: out.nam
File
Views
Analysis
out.nam
0.249369
Step: 316.2us
VIEW
0
1
2
3
4
5
6
TIME
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset
File Man...
[red-c5.t...
[root@lo...
[root@lo...
nam: ou...
Nam Co...


Applications
Places
System
10:47 AM
nam: out.nam
File
Views
Analysis
out.nam
1.298651
Step: 631.0us
VIEW
2
4
3
0
1
5
6
TIME
Auto layout:
Ca
0.15
Cr
0.75
Iterations
10
Recalc
re-layout
reset
File Man…
[red-c5.t…
[root@lo…
[root@lo…
nam: ou…
Nam Co…

FN Queue Trace Graph

```
[group@uum ]grep "a" red-queue.tr > ave.tr ; grep "Q" red-queue.tr >
            cur.tr ; cat ave.tr | awk '{print $2 " " $3}' > ave ; cat cur.tr |
            awk '{print $2 " " $3}' > cur
```



How can I add modules to NS2?

- Adding new protocol module to NS2 is possible
  - Need to create the C++ class
  - Need to create the OTcl Linkage
- More info at:
  - http://www.isi.edu/nsnam/ns/tutorial/index.html
  - Tutorial about how to add a simple protocol to NS2

NS2 Documentations

- NS2 Manual
  - Information about Otcl interpreter, C++ class hierarchy, parameters for various protocols
  - http://www.isi.edu/nsnam/ns/doc/index.html
  - Very detailed, useful when looking for something specific, like:
    - What are the shadowing models available for wireless? How do I select them?
    - How do I make my routing strategy to be Distance Vector routing?

NS2 Documentations

- NS2 Tutorial by Marc Greis
  - http://www.isi.edu/nsnam/ns/tutorial/index.html
  - Good starting point for understanding the overall structure of NS2
  - Examples:
    - What is the relation between c++ classes and Otcl classes?
    - basic info on instantiating NS2 instance, tcl scripting

NS2 Documentations

- NS2 for beginners
  - http://www-sop.inria.fr/maestro/personnel/Eitan.Altman/COURS-NS/n3.pdf
  - More detailed than Marc Greis' Tutorial
  - More info on getting it up and running – rather than internals
  - Examples:
    - What does each line of a tcl script do?
    - Most common examples of trace formats that are useful

Tcl Documentations

- Tcl Tutorial
  - http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html
- Tcl Manual
  - All commands and their explanation
  - http://www.tcl.tk/man/tcl8.6/TclCmd/contents.htm

Bug-Fixing – When things go wrong..

- Googling for the problem! ☺
  - Extensive NS2 mailing lists
  - Chances are that other people have had the same problem are very high
  - Responsive forums

Bug-Fixing – When things go wrong..

- NS2 in-built examples
  - Extensive inbuilt examples
    - “diffing” with the examples helps a lot
  - Sometimes a good idea to start from a script that does something similar

Bug-Fixing – When things go wrong..

- Taking a look at the code
  - Everyone adds to NS2 ☺
  - May not always confirm to the norms
    - IP TTL set to 32 instead of 256 ☺

Bug-Fixing Questions

- What is the expected behaviour of the network?
- Have I connected the network right?
- Am I logging trace information at the right level? Can I change it to narrow down on the problem?
- Has anyone else out there had the same problem?
- Is there something similar in examples that I can look at, and build upon?
- Does the code really do what the protocol says? Are all the default parameters correct?
- Is Tcl being picky here? ☺

Wise Words …

“Some analysts start with complex models that cannot be solved or a simulation project with very ambitious goals that are never achieved. It is better to start with simple models or experiments, get some results or insights, and then introduce the complication”

Raj Jain

Conclusion

In this session, we have:

- *Explained* network performance evaluation techniques commonly used in the computer systems and networking research
- *Identified* suitable technique(s) that can be used in specific network performance evaluation -based research
- *Described* the simulation-based network performance evaluation technique commonly used in the computer systems and networking research
- *Demonstrated* the use of NS2 for simulation-based network performance research

Thank You