Introduction NS2 Simulator

Ahmad Al Hanbali

ATER Université de Nice Sophia Antipolis
et membre du projet MAESTRO

Ahmad.al_hanbali@sophia.inria.fr

NS2

- Event driven network simulator
- Developed at VINT project collaboration: USC/ISI, Xerox PARC, LBNL, and UCB
- Free and open source:
  www.isi.edu/nsnam/ns/
- Object oriented program written C++/ OTCL
- Work in Linux, Mac, Windows

What NS2 can do?

- Simulate existing network protocols: TCP, routing, and multicast protocols over both wired and wireless networks.
- Possibility to implement and test new protocols and applications
- Current release: ns-2.30

NS2 Documentation

- NS2 home page & its mailing list: http://www.isi.edu/nsnam/ns/ or in Wikipedia format http://nsnam.isi.edu/nsnam/index.php/
- Introductory:
  - Marc Greis' Tutorial: http://www.isi.edu/nsnam/ns/tutorial/,
  - 'NS for beginners' E. Altman & T. Jimenez: http://www-sop.inria.fr/mistral/personnel/Eitan.Altman/COURS-NS/n3.pdf
  - 'NS by Example' Jae Chung & Mark Claypool: http://nile.wpi.edu/NS/
- Reference:
  - NS Manual also called "ns Notes and Documentation": http://www.isi.edu/nsnam/ns/ns-documentation.html

Course Objectives

- Learn to use NS2 to simulate TCP/IP and wireless networks
- Interpret the results
- Add new and change existing protocols in NS2

Planning and Evaluation

- 6 hours of theory + 9 hours of practice (TPs). The last TP will be marked.
- 3 hours of evaluation (exam)
- Mark=60% of highest mark + 40% of lowest mark.

Outline

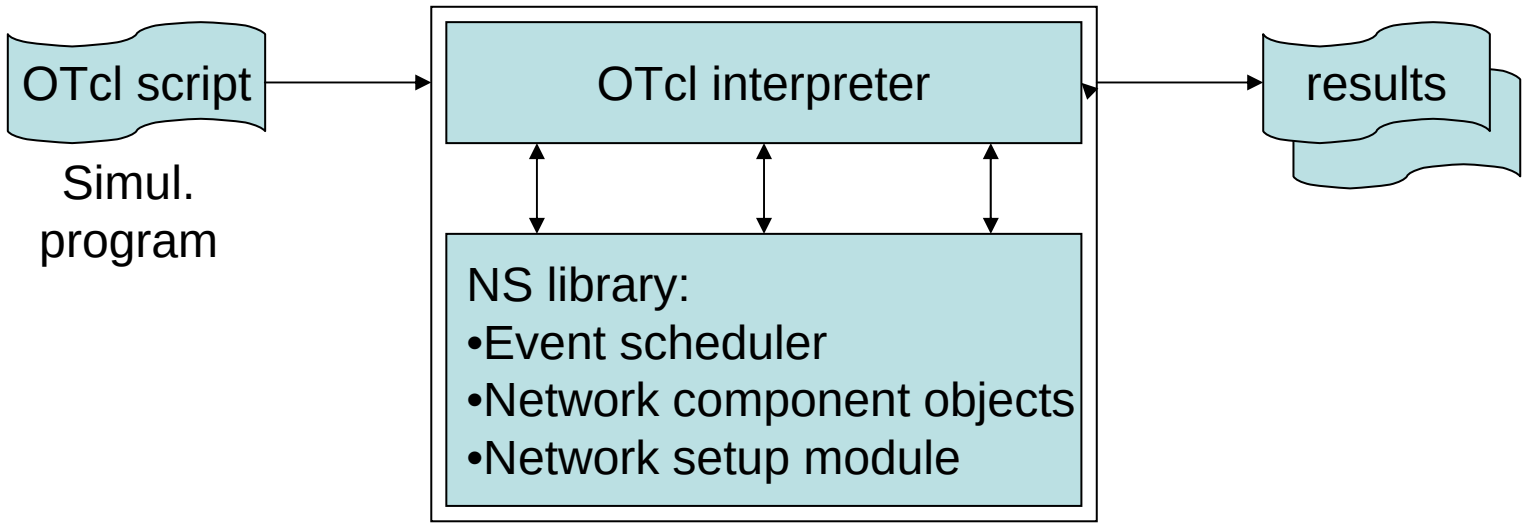
- Install NS2
- Introduction
  - Basic OTCL script
  - Networks components
  - Event scheduler
- How to simulate
  - OTCL programming
  - Create a scenario
  - Extract results: nam, plots, trace files
- Simulation run

Install NS2 in Linux

- Install all pieces at once using ns-allinone-xxx package. For details see: http://nsnam.isi.edu/nsnam/index.php/Downloading_and_installing_ns-2
- By pieces is more difficult because of pieces dependency management.

Introduction

- NS2 is OO Tcl script interpreter that has a simulation event scheduler, network component, and network setup module.



Otcl script

- The basic ns2 Otcl script should contain at least:

  1- set ns [new Simulator]; # definition of the object ns to be the simulator.

  2- $ns at simul-time “exit 0”

  3- $ns run; # to execute the simulation should be added at the end of the script

Network components classes (1)

- Application: parent class of application (ftp, telnet, traffic generator, etc)
- Agent: parent class of all protocols of transport and network layers (TCP, UDP, TFRC, RTP, RIP, OSPF, SRM, DVMRP, PIM, etc)
- Node: represents set of nodes in network. A node can be a terminal, router, or a gateway. A node contains a *classifier* to decide to where to send a packet received.
- Queue: parent class of all buffers (Drop tail, FQ, SFQ , RED, etc)
- Linkdelay: amount of time for a packet to traverse a link, e.g. a packet of size s and link with rate b and a propagation delay d will have a link delay=(s/b)+d

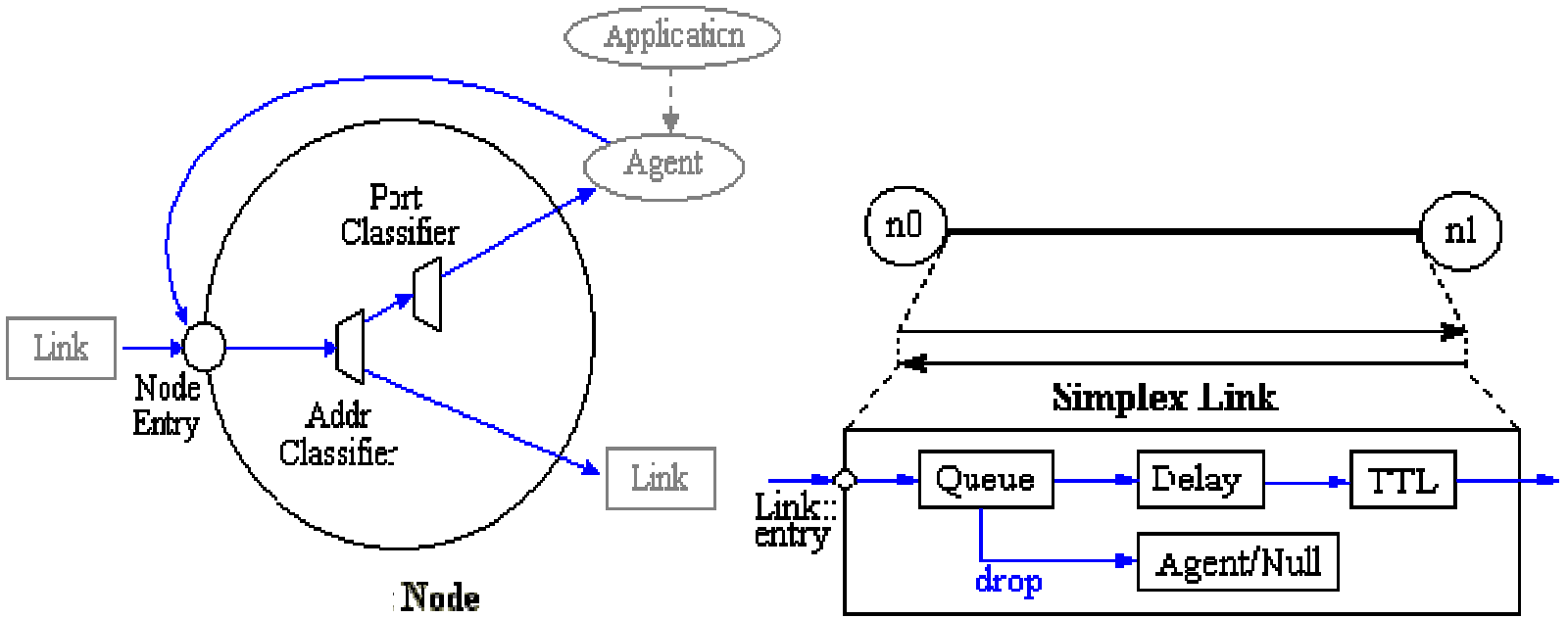
Link=Queue + Linkdelay

Network components classes (2)

- Packet: fundamental unit of exchange between network objects in simulation. It contains a packet header of the protocols implemented in NS2.
  - By default NS2 includes ALL packet headers of ALL protocols in NS2 in EVERY packet in your simulation. (for the size of all packet header in NS2 is about 2KB, if you turn on only the common header IP and TCP headers they add up to 100bytes.)
  - To remove a packet header (for example AODV) from your simulation add in the beginning of your simulation code program: remove-packet-header AODV
  - List of all packet headers in version 2.30 for example:

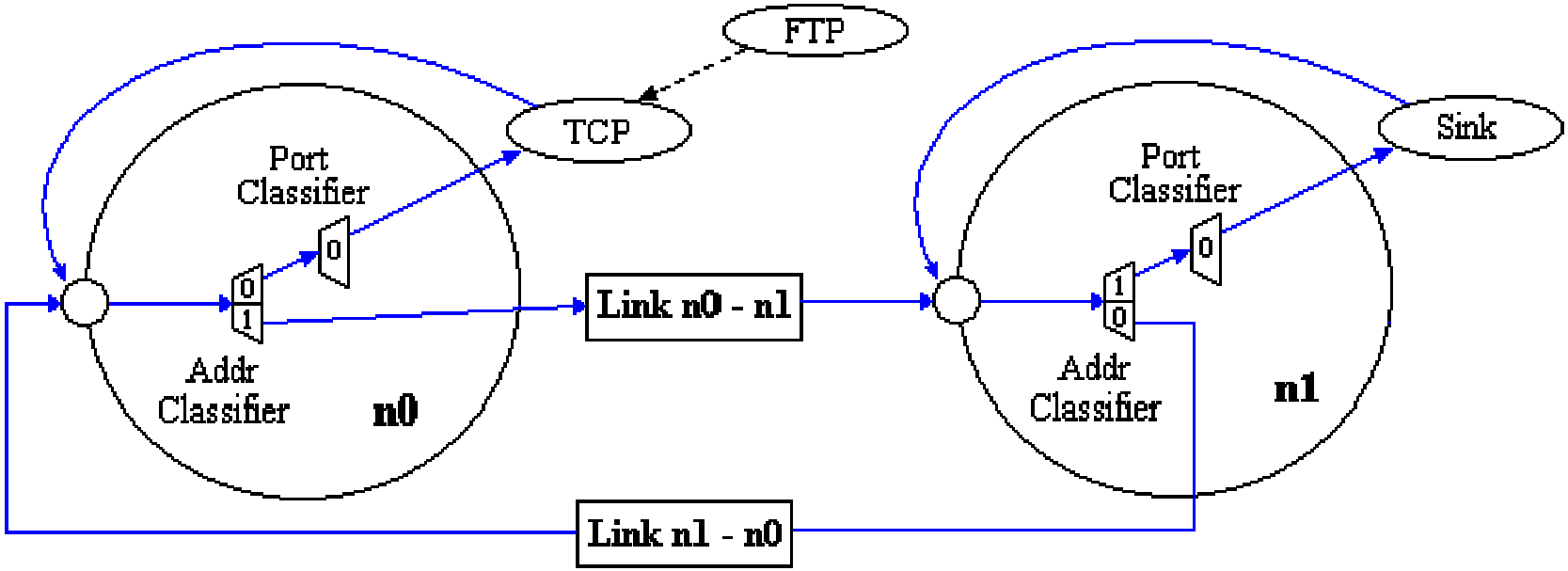
    ~/ns-allinone-2.30/ns-2.30/tcl/lib/ns-packet.tcl

Node and Link



Figures copied from
'Ns by Example' of J. Chung & M. Claypool

Example of two nodes



Figures copied from
'Ns by Example' of J. Chung & M. Claypool

Event scheduler (1)

- NS2 is an event-driven simulator. The scheduler runs by selecting the next earlier event, executing it to completion, and returning to execute the next event. Unit of time used seconds.
- Network components communicate by passing packets however this does not consume simulation time. All network components that need to spend some simulation time handling a packet (i.e. need a delay) use the event scheduler by issuing an event for the packet and waiting for event to be fired before doing further action.

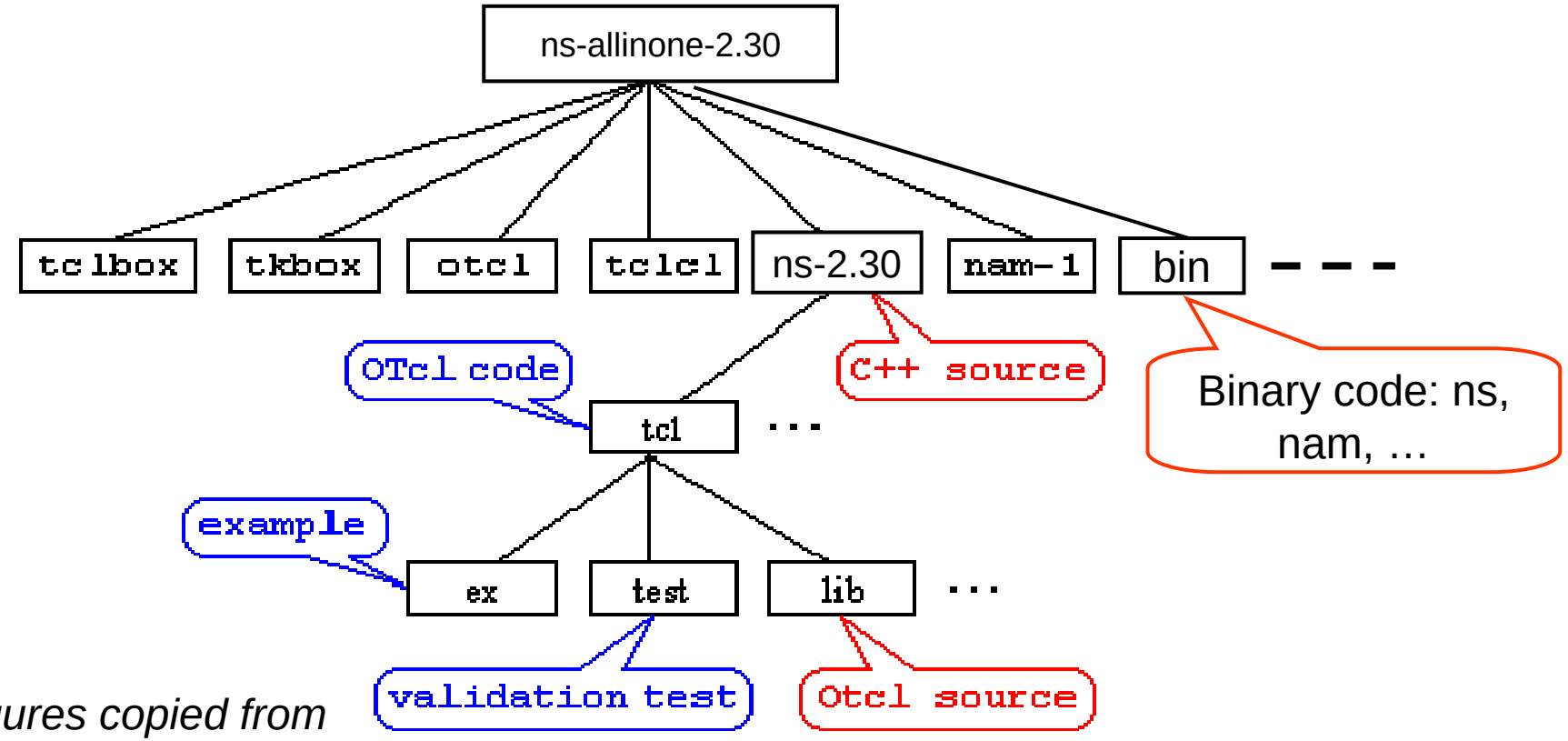
  Example: a switch with 20 microseconds switching delay issues an event for a packet to be switched to the scheduler as an event 20 microseconds later.
- Another use of event scheduler is timer.

  Example: issue a retransmission timeout if the Ack of TCP packet is not received after RTO seconds.

Event scheduler (2)

- The procedure “$ns at nn” in simulation scripts allows to execute the an Otcl procedure at time nn. For example: to call a procedure start at time 0 you may write: $ns at 0 “start”. Or to write 10 s of the simulation time has passed you may write: $ns at 10 “puts \”10 seconds has been passed\” “
- How it works:
  - In the Otcl scripts the user schedule a certain number of event to be executed during the lifetime of simulation, e.g. $ns at 10 “$ftp start 1000”
  - The network objects schedule their own event
  - All the events are added in a queue in time order manner. Scheduler start to execute the first event t when “$ns run” command is called.

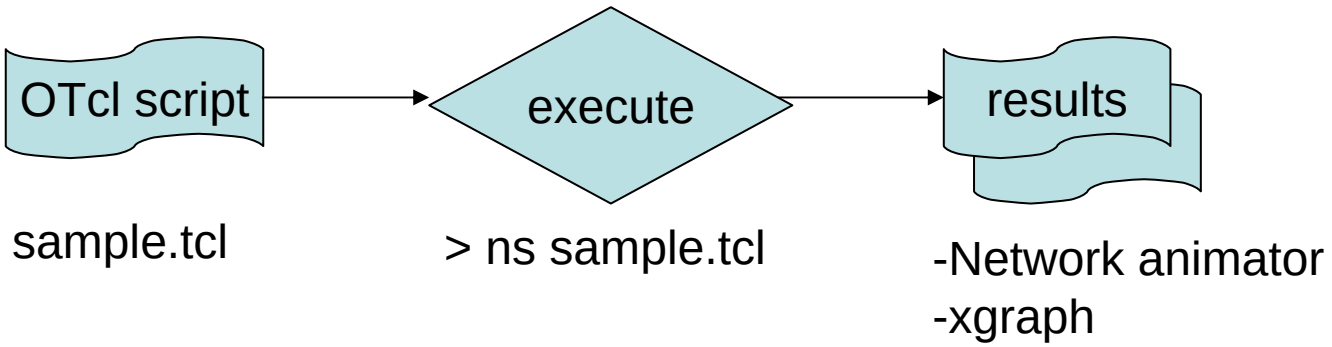
NS2 Directory Structure



Figures copied from 'Ns by Example' of J. Chung & M. Claypool

Simulation stages

1. Define the scenario to simulate: network topology, application.
2. Write the Otcl script
3. Execute the script and extract the results



Otcl programming (1)

- High level object oriented script language.
- Basic instructions:
  - Assign the value of 10 to variable *a*: set a 10
  - Line comment: #comment line
  - Assign to var. *b* the value of var. *a*: set b $a
  - Open a file *f* in write mode: set f [open file w]
  - Write the value of var. *a* in file *f*: puts $f “$a”
  - Arithmetic operation c=2*a/b: set c [expr 2.0*$a/$b]
  - Control structure: -if {$a==10} {….} else {….}
    -for {set i 1} {$i<10} {set i [expr 2*$i]} {…..}

Otcl programming (2)

- Pass the parameter 10 and 20 to the simulator:
  - » In executing command line: ns sample.tcl 10 20
  - » Inside the script: set a [lindex argv 0]; #a=10
    set b [lindex argv 1]; #b=20
- Define an array *tab* of size 10:
  - » for {set i 1} {$i<=10} {incr i} {
    set tab($i) 0; }
- Define a new function myfunc with parameters para1 and para2:
  - » proc myfunc {para1 para2} {
    global a b;
    return [expr $a/$para1+$para2] }
- Call the function myfunc: set e [myfunc $para1 $para2]

Otcl programming (3)

- Create an Otcl object of a class:
  - » set tcp1 [new Agent/TCP]; # TCP source agent
  - » set sink1 [new Agent/TCPSink]; #TCP destination agent
  - » set ftp1 [new Application/FTP]; # file transfer protocol FTP application
- Call a void function of an object:
  - » $ftp send 100
- Call a function with return:
  - » set a [$object function-name $para1]
- Assign a value to an attribute of an object:
  - » $tcp set packetSize_ 1000
- Read an attribute of an object:
  - » Set a [$tcp set packetSize_]

Create a scenario

- Create a simulation object: *set ns [new Simulator]*
- Open a file for writing to be used for nam (network animator) trace data:

```
set nf [open out.nam w] #nf is the handle of file out.nam
$ns namtrace-all $nf  # tell the simulator to put all the events
                      # relevant to simulator into this file
set trace [open trace.tr w] # open trace.tr file in writing mode
$ns trace-all $trace # tell the simulator to put all the trace
                     # event in trace file
```

Create a scenario

- Write the finish procedure that will close all the trace and the nam:

```
» proc finish { } {
    global ns nf trace
    $ns flush-trace
    close $nf # close the network animator file
    close $trace # close the trace file
    exec nam out.nam & exit 0 # call the animator &
                                  #  exit
}
```

Create a scenario

- Create the two nodes and setup the link between them:
  - » set n0 [$ns node] # create the first node
  - » set n1 [$ns node] # create the second node
  - » $ns duplex-link $n0 $n1 1Mb 10ms Droptail
  - » # create the link between these 2 nodes of bandwidth 1Megabits per seconds (Mbps), on link delay 10 ms, and of type Droptail.
- Create the transport agent and the source traffic:
  - » set udp0 [new Agent/UDP] # create the udp agent
  - » $ns attach-agent $n0 $udp0 # attach it to node 0
  - » set cbr0 [new Application/Traffic/CBR]
  - » $cbr0 set packetSize_ 500 # the packet size in bytes
  - » $cbr0 set interval_ 0.005 # cbr0 will a send a packet every 5ms
  - » $cbr0 attach-agent $udp0 # attach the udp0 to node 0

Create a scenario

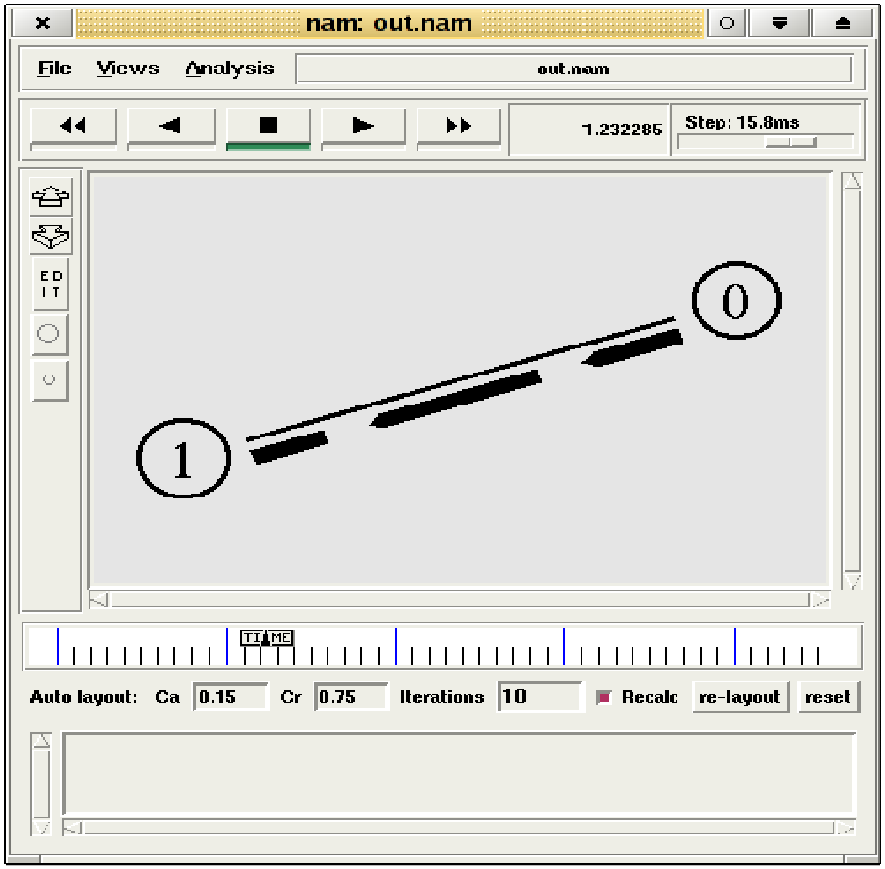
- Create the null agent and attach it to node 1:
  - » set null0 [new Agent/Null]
  - » $ns attach-agent $n1 $null0 # attach the null agent to node 1
- Connect the transport agents UDP and Null:
  - » $ns connect $udp0 $null0

Create a scenario

- Make the CBR traffic source to start to send data at 0.5sec. and to stop at 4.5sec.:
  - » $ns at 0.5 “$cbr0 start”
  - » $ns at 4.5 “$cbr0 stop”
- Call the finish function before running the simulation:
  - » $ns at 5.0 “finish”
  - » $ns run

Scenario outputs: Nam

- exec nam out.nam



Scenario outputs: Trace

- Trace file trace.tr :

r	0.514	0	1	cbr	500	----	0	0.0	0.1	0	1
Event	Event time	Link btw node 0 &1		Pkt type	Size bytes	flag	Flow id	Src addr of pkt	Dst addr	Seq no	Pkt id

- Events: + pkt entered link queue , - pkt dequeued from link queue, r pkt received by its destination, d pkt has been dropped.
- Open file trace.tr

Trace.tr

```
+ 0.5 0 1 cbr 500 ------- 0 0.0 2.0 0 0
- 0.5 0 1 cbr 500 ------- 0 0.0 2.0 0 0
+ 0.505 0 1 cbr 500 ------- 0 0.0 2.0 1 1
- 0.508 0 1 cbr 500 ------- 0 0.0 2.0 1 1
+ 0.51 0 1 cbr 500 ------- 0 0.0 2.0 2 2
+ 0.515 0 1 cbr 500 ------- 0 0.0 2.0 3 3
- 0.516 0 1 cbr 500 ------- 0 0.0 2.0 2 2
r 0.518 0 1 cbr 500 ------- 0 0.0 2.0 0 0
+ 0.518 1 2 cbr 500 ------- 0 0.0 2.0 0 0
- 0.518 1 2 cbr 500 ------- 0 0.0 2.0 0 0
+ 0.52 0 1 cbr 500 ------- 0 0.0 2.0 4 4
```

Scenario outputs: Extract result from simulation

- At any time of simulation you can read the value of any variable of the network object write it to the screen or put it into a file. For example to print out on the screen the congestion window size of TCP session:
  - » $ns at time “puts [$tcp set cwnd_]”

Extract data from trace file (1)

- Use *awk* or *perl* scripting language to interpret the trace file
- For awk to extract data from trace file trace.tr.
- Write the awk script (that is similar to C language) and save it in script.awk:
  - » BEGIN { initialize variables; }
    { for each line of the trace instructions here will be executed; $nn returns the variable in column nn of trace; }
    END{ print out the data; }
- In shell command execute the script:
  - » awk –v var1=val1 –v var2=val2 –f script.awk trace.tr

Extract results from trace file (2)

- Example compute totsent, total number of bytes sent by node 0, and totrecv, total number of bytes received by node 1:
  - BEGIN { totsent=0; totrecv=0; }
  - {
  - if ($1==“+”) { totsent=totsent+$6; }
  - if ($1==“r”) { totrecv=totrecv+$6; }
  - }
  - END { print “Bytes sent by node 0:” totsent”\nBytes received by node 1: ” totrecv; }
- Save the script in tot.awk file and run the shell command line:
  - awk –f tot.awk trace.tr
- Press return:
  - Bytes sent by node 0: 801000
  - Bytes received by node 1: 801000

Generate graphics (1)

- Use xgraph of the NS2 simulator or Gnuplot of GNU.
- Gnuplot used in research because it support postscript format of figures used in Latex.
- You have a data text file *file.dat* that contains two columns: The first is the time. The second is the Throughput.

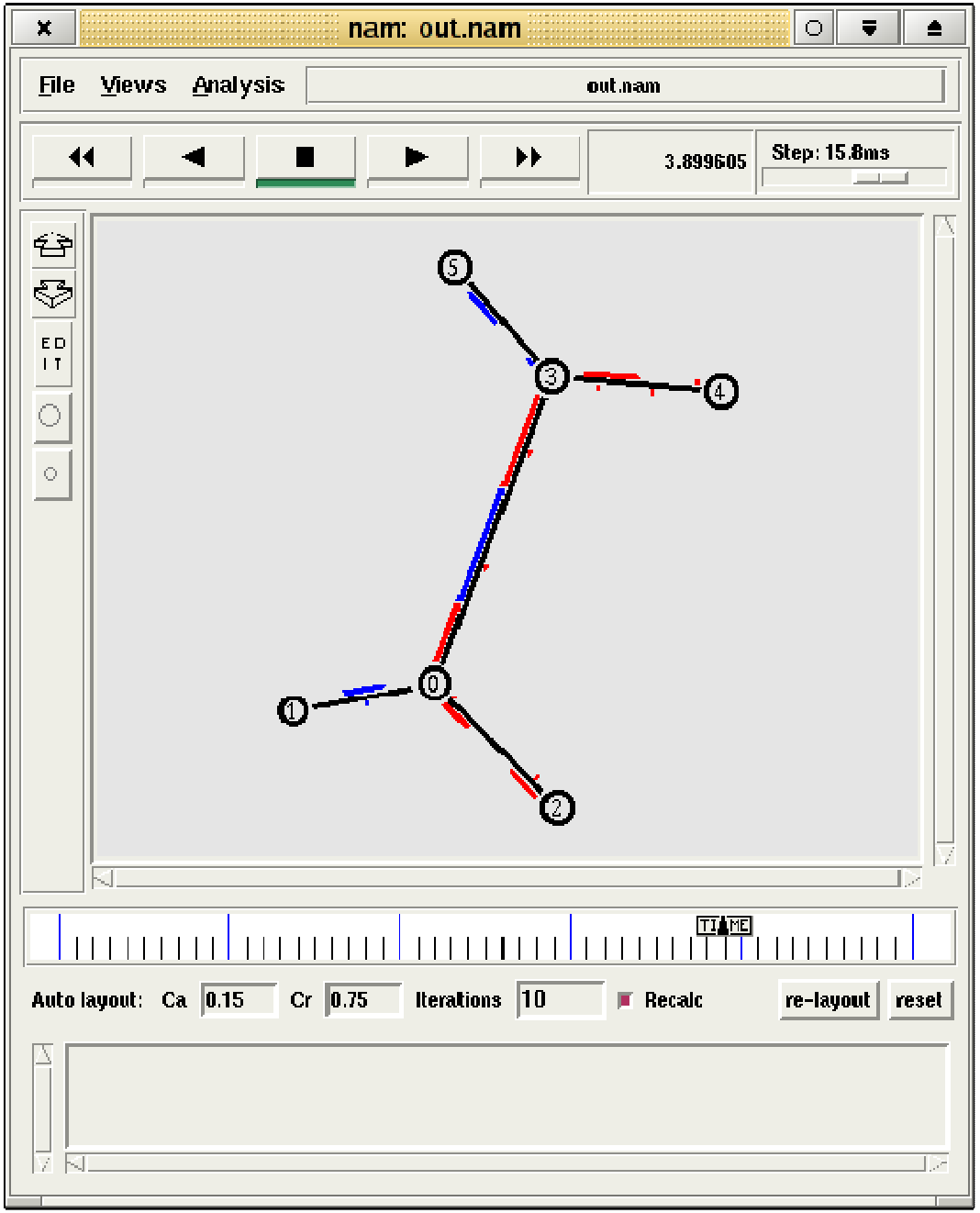
Generate graphics (2)

- Write the script and save it in comman.plot:
  - » set size 1,0.5
  - » set pointsize 1
  - » set grid lw .5
  - » set xlabel ' Temps en seconde '
  - » set label ' Taille du fenetre de congestion en paquets '
  - » set xrange [0:5]
  - » set yrange [0:50]
  - » plot 'WindowVsTime' using 1:2 title "wnd 1" w lines 1
  - » set term pos por col
  - » set out "wind.ps"
  - » replot
- Execute in shell: gnuplot command.plot and open the figure wind.ps in postscript (gv wind.ps).

Demonstration: TCP Scenarios

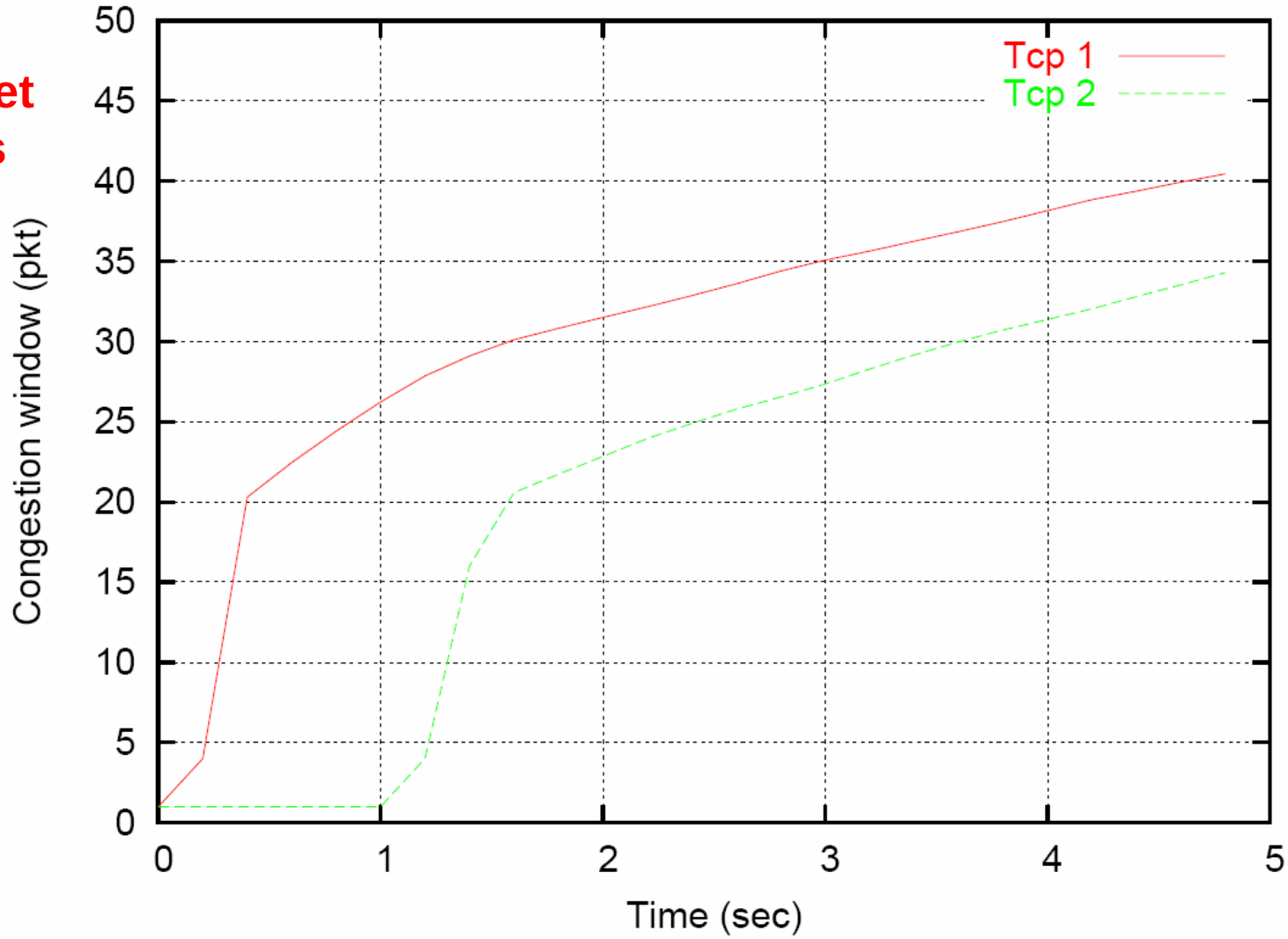
Scenario 1

- One tcp session between nodes 1 and 5 that starts at time 0s
- One tcp session between node 2 and 4 that starts at time 1s
- Simulation Duration 5s



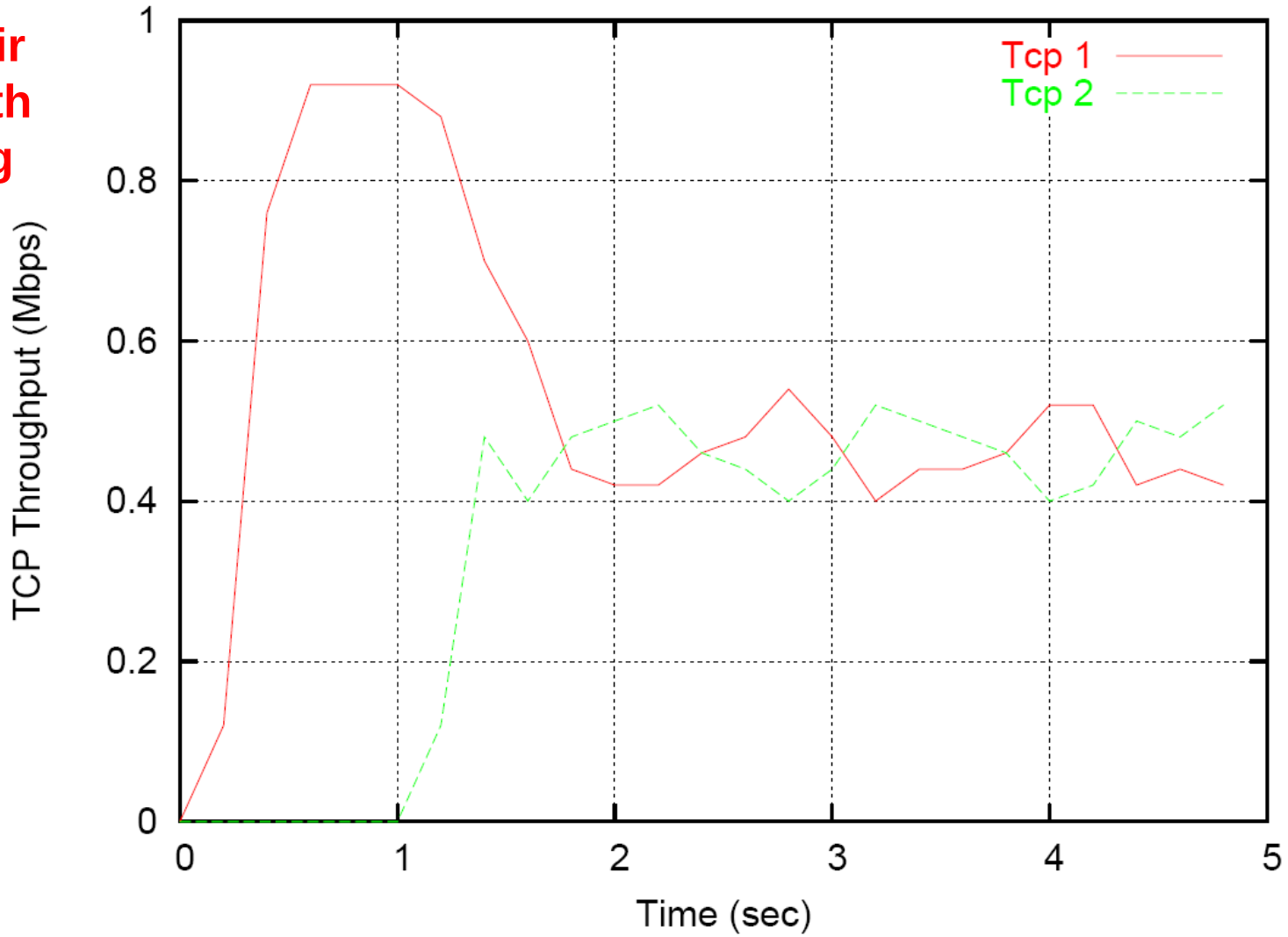
Congestion Window

No packet loss



Throughput

TCP Fair bandwith sharing

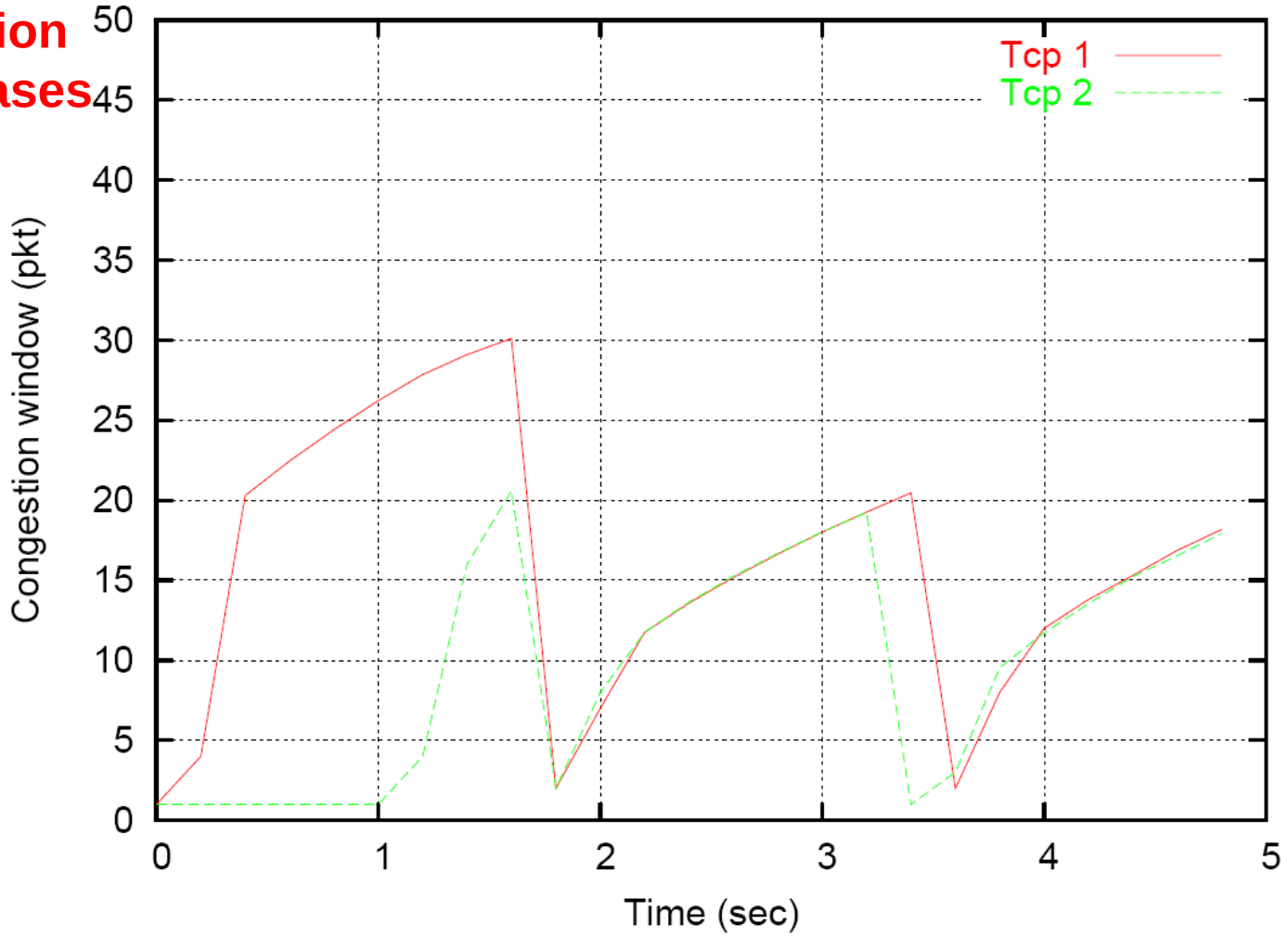


Scenario 2

- Introduce congestion by limiting the maximum queue size between node 0 and 3 to 20 packets
- $ns queue-limit $n(0) $n(3) 20

Congestion window

When congestion window decreases TCP detects losses



Throughput

On loss TCP decreases it throughput