ns-2 Tutorial Exercise

Multimedia Networking Group,
The Department of Computer Science,
UVA
Jianping Wang

Partly adopted from Nicolas's slides

On to the Tutorial…

- Work in group of two. At least one people in each group must have an account on the CS department UNIX servers.
- Start Exceed on the machine at your desk. Login on one of the CS compute servers (mamba.cs, viper.cs, etc).
- Set up your environment so that you can use ns, by issuing the following commands:

  export PATH=$PATH:/home/cs757/ns/ns:/home/cs757/ns/nam

  export PATH=$PATH:/home/cs757/ns/xgraph
- Create a temporary subdirectory at current directory:

  mkdir ns2-tutorial-temp

  cd ns2-tutorial-temp

  cp /home/cs757/ns-tutorial/*.tcl .
- Scripts can be found in /home/cs757/ns-tutorial.

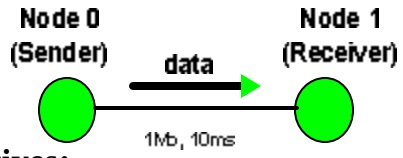
Exercise Tutorial Covered

- **Marc Greis' ns tutorial can be found at:**
  http://www.isi.edu/nsnam/ns/tutorial/index.html
- **Chapter IV: The first Tcl script**
- **Chapter V: Making it more interesting**
- **A TCP example**
- **Chapter VI: Network Dynamics**
- **Chapter VIII: Creating Output Files for *xgraph***

The first Tcl Script

- **Create a script that simulates the simplest topology:**

- **example1a.tcl only builds network topology.**
- **example1b.tcl creats traffic from Node 0 to Node 1.**



- **Objectives:**
  - **Get a basic understanding of the way objectives interact in ns.**
  - **Lay the foundations for more complicated simulations.**

Example1a.tcl

```
#Create a 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 –a out.nam &
        exit 0
}
```

Example1a.tcl (cont')

```
#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

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

#Run the simulation
$ns run
```

Execute a ns-2 script:

- ns example1a.tcl

Example1b.tcl

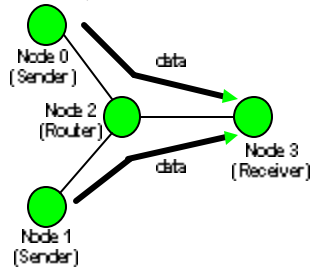
```
#Create a simulator object
set ns [new Simulator]

#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 –a out.nam &
        exit 0
}

#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
```

Example1b.tcl (cont')

```
#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 with the traffic sink
$ns connect $udp0 $null0

#Schedule events for the CBR agent
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Making it more interesting

- **Build on the first script to simulate a simple network (example2.tcl).**



- **Objectives:**
  - **Be introduced to some of the commands that can be passed to nam.**
  - **Get a basic understanding of the network parameters that can be modified.**

Example2.tcl

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

#Define different colors for data flows
$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 trace file
        close $nf
    #Execute nam on the trace file
        exec nam –a out.nam &
        exit 0
}

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

Example2.tcl (cont' 1)

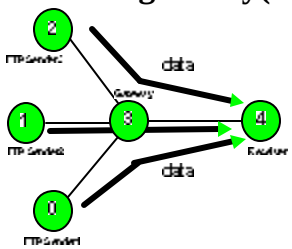
```
#Create links between the nodes
$ns duplex-link $n0 $n2 1Mb 10ms  DropTail
$ns duplex-link $n1 $n2 1Mb 10ms  DropTail
$ns duplex-link $n3 $n2 1Mb 10ms SFQ

$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 the link between node 2 and node 3
$ns duplex-link-op $n2 $n3 queuePos 0.5

#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$udp0 set class_ 1
$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 UDP agent and attach it to node n1
set udp1 [new Agent/UDP]
$udp1 set class_ 2
$ns attach-agent $n1 $udp1
```

Example2.tcl (cont' 2)

```
# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1

#Create a Null agent (a traffic sink) and attach it to node n3
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0

#Connect the traffic sources with the traffic sink
$ns connect $udp0 $null0
$ns connect $udp1 $null0

#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 1.0 "$cbr1 start"
$ns at 4.0 "$cbr1 stop"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

A TCP example

- **Create a simple TCP scenario with droptail queue mechanism on the gateway(example3.tcl).**



- **Objectives:**
  - **Understand the basic components of TCP transmission.**
  - **Observe the behavior of TCP sessions.**

Example3.tcl

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

#Open trace files
set f  [open droptail-queue-out.tr w]
$ns trace-all $f

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


#s1, s2 and s3 act as sources.
set s1 [$ns node]
set s2 [$ns node]
set s3 [$ns node]

#G acts as a gateway.
set G  [$ns node]

#r acts as a receiver.
set r [$ns node]

#Define different colors for dats flows
$ns color 1 red    ;# the color of packets from s1
$ns color 2 SeaGreen  ;# the color of packets from s2
$ns color 3 blue   ;# the color of packets from s3
```

Example3.tcl (cont' 1)

```
#Create links between the nodes
$ns duplex-link $s1 $G  6Mb 1ms DropTail
$ns duplex-link $s2 $G  6Mb 1ms DropTail
$ns duplex-link $s3 $G  6Mb 1ms DropTail
$ns duplex-link $G  $r  3Mb 1ms  DropTail

#Define the queue size for the link between node G and r
$ns queue-limit $G $r 5

#Define the layout of the topology
$ns duplex-link-op $s1 $G orient right-up
$ns duplex-link-op $s2 $G orient right
$ns duplex-link-op $s3 $G orient right-down
$ns duplex-link-op $G  $r orient right

#Monitor the queues for links
$ns duplex-link-op $s1 $G  queuePos 0.5
$ns duplex-link-op $s2 $G  queuePos 0.5
$ns duplex-link-op $s3 $G  queuePos 0.5
$ns duplex-link-op $G  $r queuePos 0.5

#Create a TCP agent and attach it to node s1
set   tcp1 [new Agent/TCP/Reno]
$ns   attach-agent $s1 $tcp1
$tcp1 set window_ 8
$tcp1 set fid_    1
```

Example3.tcl (cont' 2)

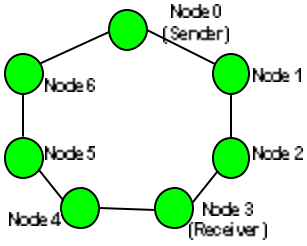
```
#Create a TCP agent and attach it to node s2
set tcp2 [new Agent/TCP/Reno]
$ns   attach-agent $s2 $tcp2
$tcp2 set window_ 8
$tcp2 set fid_    2

#Create a TCP agent and attach it to node s3
set tcp3 [new Agent/TCP/Reno]
$ns attach-agent $s3 $tcp3
$tcp3 set window_ 4
$tcp3 set fid_    3

#Create TCP sink agents and attach them to node r
set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]
set sink3 [new Agent/TCPSink]
$ns attach-agent $r $sink1
$ns attach-agent $r $sink2
$ns attach-agent $r $sink3

#Connect the traffic sources with the traffic sinks
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
$ns connect $tcp3 $sink3

#Create FTP applications and attach them to agents
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
```

Example3.tcl (cont' 3)

```
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3

#Define a 'finish' procedure
proc finish {} {
global ns
   $ns flush-trace
puts "running nam..."
   exec nam –a droptail-queue-out.nam &
   exit 0
}

$ns at 0.0   "$s1 label Sender1"
$ns at 0.0   "$s2 label Sender2"
$ns at 0.0   "$s3 label Sender3"
$ns at 0.0   "$G  label Gateway"
$ns at 0.0   "$r  label Receiver"
$ns at 0.1   "$ftp1 start"
$ns at 0.1  "$ftp2 start"
$ns at 0.1   "$ftp3 start"
$ns at 5.0   "$ftp1 stop"
$ns at 5.0   "$ftp2 stop"
$ns at 5.0   "$ftp3 stop"
$ns at 5.25  "finish"

$ns run
```

Network Dynamics

- **Create a more complex topology, simulate a link failure (example4.tcl).**



- **Objectives:**
  - **Exposed to more complex oTcl constructs (e.g., *for* loops).**
  - **Exposed to types of events different from "start" and "stop".**
  - **Observe a DV routing protocol in action.**

Example4.tcl

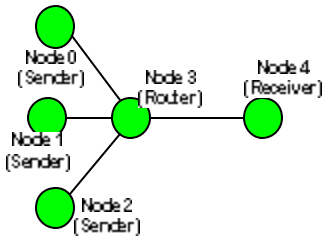
```
#Create a simulator object
set ns [new Simulator]

#Tell the simulator to use dynamic routing
$ns rtproto DV

#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 –a out.nam &
        exit 0
}

#Create seven nodes
for {set i 0} {$i < 7} {incr i} {
        set n($i) [$ns node]
}
```

Example4.tcl (cont')

```
#Create links between the nodes
for {set i 0} {$i < 7} {incr i} {
      $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail
}

#Create a UDP agent and attach it to node n(0)
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $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 n(3)
set null0 [new Agent/Null]
$ns attach-agent $n(3) $null0

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

#Schedule events for the CBR agent and the network dynamics
$ns at 0.5 "$cbr0 start"
$ns rtmodel-at 1.0 down $n(1) $n(2)
$ns rtmodel-at 2.0 up $n(1) $n(2)
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

Creating Output Graphs for *xgraph*

- **Plot the throughput obtained by different flows in the following network(example5.tcl):**



- **Objectives: - Be exposed to more diverse traffic generators.**
  - Learn the basics on how to postprocess a simulation to get useful information.

Example5.tcl

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

#Open the output files
set f0 [open out0.tr w]
set f1 [open out1.tr w]
set f2 [open out2.tr w]

#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

#Connect the nodes
$ns duplex-link $n0 $n3 1Mb 100ms DropTail
$ns duplex-link $n1 $n3 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 1Mb 100ms DropTail
$ns duplex-link $n3 $n4 1Mb 100ms DropTail

#Define a 'finish' procedure
proc finish {} {
    global f0 f1 f2
    #Close the output files
    close $f0
    close $f1
    close $f2
    #Call xgraph to display the results
    exec xgraph out0.tr out1.tr out2.tr -geometry 800x400 &
      exit 0
}
```

Example5.tcl (cont' 1)

```
#Define a procedure that attaches a UDP agent to a previously created node
#'node' and attaches an Expoo traffic generator to the agent with the
#characteristic values 'size' for packet size 'burst' for burst time,
#'idle' for idle time and 'rate' for burst peak rate. The procedure connects
#the source with the previously defined traffic sink 'sink' and returns the
#source object.
proc attach-expoo-traffic { node sink size burst idle rate } {
    #Get an instance of the simulator
    set ns [Simulator instance]

    #Create a UDP agent and attach it to the node
    set source [new Agent/UDP]
    $ns attach-agent $node $source

    #Create an Expoo traffic agent and set its configuration parameters
    set traffic [new Application/Traffic/Exponential]
    $traffic set packetSize_ $size
    $traffic set burst_time_ $burst
    $traffic set idle_time_ $idle
    $traffic set rate_ $rate

      # Attach traffic source to the traffic generator
      $traffic attach-agent $source
    #Connect the source and the sink
    $ns connect $source $sink
    return $traffic
}
```

Example5.tcl (cont' 2)

```
#Define a procedure which periodically records the bandwidth received by the
#three traffic sinks sink0/1/2 and writes it to the three files f0/1/2.
proc record {} {
    global sink0 sink1 sink2 f0 f1 f2
  #Get an instance of the simulator
  set ns [Simulator instance]
  #Set the time after which the procedure should be called again
    set time 0.5
  #How many bytes have been received by the traffic sinks?
    set bw0 [$sink0 set bytes_]
    set bw1 [$sink1 set bytes_]
    set bw2 [$sink2 set bytes_]
  #Get the current time
    set now [$ns now]
  #Calculate the bandwidth (in MBit/s) and write it to the files
    puts $f0 "$now [expr $bw0/$time*8/1000000]"
    puts $f1 "$now [expr $bw1/$time*8/1000000]"
    puts $f2 "$now [expr $bw2/$time*8/1000000]"
  #Reset the bytes_ values on the traffic sinks
    $sink0 set bytes_ 0
    $sink1 set bytes_ 0
    $sink2 set bytes_ 0
  #Re-schedule the procedure
    $ns at [expr $now+$time] "record"
}
```

Example5.tcl (cont' 3)

```
#Create three traffic sinks and attach them to the node n4
set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
$ns attach-agent $n4 $sink0
$ns attach-agent $n4 $sink1
$ns attach-agent $n4 $sink2

#Create three traffic sources
set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]
set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]
set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]

#Start logging the received bandwidth
$ns at 0.0 "record"
#Start the traffic sources
$ns at 10.0 "$source0 start"
$ns at 10.0 "$source1 start"
$ns at 10.0 "$source2 start"
#Stop the traffic sources
$ns at 50.0 "$source0 stop"
$ns at 50.0 "$source1 stop"
$ns at 50.0 "$source2 stop"
#Call the finish procedure after 60 seconds simulation time
$ns at 60.0 "finish"

#Run the simulation
$ns run
```