NS2 Tutorial

Chang-Gun Lee (cglee@snu.ac.kr)

Assistant Professor

The School of Computer Science and Engineering

Seoul National University

What we need

- Linux (e.g., RedHat 8.0) installed with full options
- Tcl/Tk (version 8.4.14)
- otcl (version 1.12)
- TclCL (version 1.18)
- ns-2 (version 2.30)
- nam (version 1.11)
- Xgraph (version 12.1)
- Get all of them (piece by piece) from www.isi.nsnam/ns (click “Download and Build ns)

Step 1: Tcl/Tk install

1. Already done!

Step 2: otcl install

1. Decompress
   - tar zxvf otcl-src-1.12.tar.gz
2. Configure
   - cd otcl-1.12
   - ./configure --with-tcl=/usr/X11R6 --with-tcl-ver=8.4 --with-tk=/usr/X11R6
   - If it complains that “tclInt.h” doesn’t exist”, copy tclInt*.h from tcl8.4.14/generic to /usr/X11R6/include
3. Make
   - cd otcl-1.12
   - make (this will creat otclsh, owish, libotcl.a, libotcl.so)
4. Test
   - cd otcl-1.12
   - make test (skip!)
5. Install (be the super user first)
   - cd otcl-1.12
   - make install (this will copy files to /usr/local/bin, /usr/local/lib, /usr/local/include)

Step 3: TclCL install

1. Decompress
   - tar zxvf tclcl-src-1.18.tar.gz
2. Configure
   - cd tclcl-1.18
   - ./configure --with-tcl=/usr/X11R6 --with-tcl-ver=8.4 --with-tk=/usr/X11R6 --with-tk-ver=8.4
3. Make
   - cd tclcl-1.18
   - make (this will creat tcl2c++, libtclcl.a)
4. Install (be the super user first)
   - cd tclcl-1.18
   - make install (this will copy files to /usr/local/bin, /usr/local/lib, /usr/local/include)

Step 4: NS-2 install

1. Decompress
   - tar zxvf ns-src-2.30.tar.gz
2. Configure
   - cd ns-2.30
   - ./configure --with-tcl=/usr/X11R6 --with-tcl-ver=8.4 --with-tk=/usr/X11R6 --with-tk-ver=8.4
3. Make
   - cd ns-2.30
   - make
4. Test
   - cd ns-2.30
   - ./validate
   - Note: if it complains that “Cannot load libotcl8.4.so”, copy otcl-1.12/libotcl.so to /usr/local/lib and add “usr/local/lib” to LD_LIBRARY_PATH by editing .bashrc
5. Install
   - Copy ns to /usr/local/bin

Step 5: nam install

1. Get the binary release nam-1.11-linux-i386.tar.gz from http://www.isi.edu/nsnam/nam/index.html
2. Decompress
   - tar xzvf nam-1.11-linux-i386.tar.gz
3. Install
   - copy nam /usr/local/bin

Step 6: Xgraph install

1. Decompress
   - tar xzvf xgraph-12.1.tar.gz
2. Configure
   - cd xgraph-12.1
   - ./configure
3. Make
   - cd xgraph-12.1
   - make
4. Install (be the super user)
   - cd xgraph-12.1
   - make install (this will copy xgraph to /usr/local/bin)

NS-2 Tutorial (1)

- Two nodes connected through a duplex link
  - Source node send CBR traffic over UDP during the time interval [0.5 sec, 4.5 sec]
  - Destination node receive it
- What we program
  - example1.tcl: specify node topology and simulation scenario
- What we have to do
  - ns example1.tcl
- What to learn
  - How to define node and their connections
  - How to use the existing protocol agents (e.g., UDP)
  - How to use the application agents (e.g., CBR)
  - How to run “nam” in the tcl script to view the simulation

example1.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 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
```

```
#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
```

NS-2 Tutorial (2)

- Three nodes connected through duplex links
  - Source node (Node 0) sends CBR traffic over UDP during the time interval [0.5 sec, 4.5 sec] via Node 2 toward the final destination Node 3
  - Source node (Node 1) sends CBR traffic over UDP during the time interval [1.0 sec, 4.0 sec] via Node 2 toward the final destination Node 3
  - Router (Node 2) routes the traffic
  - Destination node (Node 3) receive it
- How to run
  - ns example2.tcl
- What to learn
  - How to classify flows (visualize them with different colors)
  - How to monitor a queue
  - Observe unfair drop by DropTail queue
  - Observe if SFQ (Stochastic Fair Queueing) can solve the unfairness

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 {} {
        …. same as before ….
}

#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 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 // byte
$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

```
# 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
```

NS-2 Tutorial (3)

- A ring topology with 7 nodes
  - Source node (Node 0) send CBR traffic over UDP during [0.5 sec, 4.5 sec] toward Node 3
  - Destination node (Node 3) receives it
  - All other nodes work as routers
  - Link between Node 1 and Node 2 downs at 1.0 sec and recovers at 2.0 sec
- How to run
  - ns example3.tcl
- What to learn
  - How to use node “array” and for loop in tcl script to model many nodes
  - How to simulate link failure
  - Observe all packets drop while the link fails
  - How to use DV (Distance Vector) routing
  - Observe if DV finds another path detouring the failed link

example3.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 {} {
 … same as before ….
}

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

NS-2 Tutorial (4)

- Three source nodes (Nodes 0, 1, 2) send burst traffic via a router (Node 3) toward the final destination (Node 4)
  - Source nodes (Nodes 0, 1, 2) start and stop the burst traffic at 10 sec and 50 sec, respetively.
  - Three sinks at the Destination node (Node 4) records the bandwidth of three flows at every 0.5 sec.
  - The recorded bandwidth (out0.tr, out1.tr, and out2.tr) is displayed by Xgraph
- How to run
  - ns example4.tcl
- What to learn
  - How to generate burst traffic
  - How to record the simulated data into files
  - How to run Xgraph to visualize the recorded data

example4.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 $f0
  close $f1
  close $f2
  exec xgraph out0.tr out1.tr out2.tr -geometry 800x400 &
  exit 0
}
```

```
#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
}
```

example4.tcl

```
#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"
}
```

```
#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
$ns at 60.0 "finish"

#Run the simulation
$ns run
```