Introduction to exercise 3

Using ns2

- Exercise 3 will be carried on using the ns2 object oriented discrete event network simulator.
- To use ns2, connect yourself to a workstation, where ns2 is installed. For instance any computer in Maari A. (www.hut.fi/cc/computers/Maari-A.html)
- To load the program to your own path, give the command

  `source /p/edu/s-38.180/use.csh`
  (The warning message `DISPLAY: Undefined variable` can be disregarded.)
- Run the simulation file saved as, for instance, filename.tcl with the command

  `ns filename.tcl`
  (There might be a warning message about perl versions when running ns2, but you can disregard that. It does not affect these exercises)
- There are ns2 documentations and tutorials available, of which this document is a simplified presentation.
  - http://www.isi.edu/nsnam/ns/
    - http://www.isi.edu/nsnam/ns/ns-documentation.html
    - http://www.isi.edu/nsnam/ns/tutorial/index.html
  - http://nile.wpi.edu/NS/
- In this work, and for most simple simulations, to use ns2 it is enough to program in OTcl script language. To setup and run the simulation, a user should write an OTcl script that initiates an event scheduler, sets up the network topology using the network objects and the network setup functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler.

1 Creating the simulation code

There are example codes available on the course pages that you can use, and only make the changes required by the exercises. In order to understand the simulation so that you are able to make the necessary modifications, you have to study how the code works. The example code is presented step by step in the following sections.

Simulator basics

The lines beginning with # are comment lines and do not affect the code.

Some useful commands:

- `set`: assign a value to a variable
- `puts $filename "$variable"` : prints out the resulting operations to file, if filename omitted prints to screen
- [`expr`]to make the interpreter calculate the value of expression within the bracket after the keyword `expr`
- `proc`: to define a procedure

First we create our simulator object, which we name `nssim`

```
set nssim [new Simulator]
```

We then need to schedule some events for ns2. For now, let us schedule the end of the simulation (halt is a reserved word). We do this in the simulator object `nssim`.

```
$nssim at $simtime "$nssim halt"
```

We of course need to set the parameter simtime to some value and we will also print it on the screen.

```
set simtime 5
puts "simtime = $simtime"
```

Start the simulation

```
$nssim run
```

The last line should read

```
exit 0
```

So we have the following code

```
set simtime 5
puts "simtime = $simtime"

set nssim [new Simulator]

$nssim at $simtime "$nssim halt"

$nssim run

exit 0
```

Topology

Now we want to add some topology to the system. Let us have four nodes is a tree topology.

We add the four nodes by creating node objects and assigning them to the handles na1, na2, nb1 and nb2 using the command `set` in the following way:

```
set na1 [$nssim node]
set na2 [$nssim node]
set nb1 [$nssim node]
set nb2 [$nssim node]
```

Then we connect the nodes with links by creating a duplex-link object with the following command

```
$nssim duplex-link endpoint1 endpoint2 bandwidth delay queue-type
```

The link between na1 – nb1 and na2 – nb1 are access links, the link between nb1 and nb2 is the bottleneck link.

```
$nssim duplex-link $na1 $nb1 $abw $adel DropTail
$nssim duplex-link $na2 $nb1 $abw $adel DropTail
$nssim duplex-link $nb1 $nb2 $bnbw $bndel DropTail
```

We must also specify the queue sizes at the routers of the link.

```
$nssim queue-limit $na1 $nb1 $aqsize
$nssim queue-limit $na2 $nb1 $aqsize
$nssim queue-limit $nb1 $nb2 $bnqsize
```

Above we have used variables for the bandwidth, delay and queue size of the links. To set these variables to some values

```
set abw 10Mb
set adel 0.05
set aqsize 1000

set bnbw 2000000
set bndel 0.005
set bnqsize 50
```

The sources and sinks

With the topology in place we can now create the TCP and UDP agents. We will use the UDP traffic as background traffic filling the bottleneck link to a load of 0.6. In ns we need a sending agent, i.e. the source, and a receiving agent, i.e. the sink.

TCP agent

The aim of the exercise is to study and understand the different mechanisms built into TCP. To get better acquainted with TCP we will use the TCP RFC793edu-agent, which gives us the possibility to study the building blocks of the TCP congestion control mechanisms such as

- additive increase and multiplicative decrease (window increased by one segment per RTT, upon packet loss a time out, retransmission and window is halved)
- slow start (window initially increased with exponential speed, upon packet loss window is set to one, a time out, retransmission and then window is increased exponentially up to half and from there on window increased one per RTT)
- slow start and fast retransmit (same as above, but if enough duplicate ACKs arrive lost packet can be retransmitted without waiting for time out)

Below are the commands needed to create the TCP/RFC793edu agent.

```
# General parameters for RFC793edu - agent
set initialtimeout 5
set rto 60
set syn true
set jacobsonrtt false
set karnrtt true
set expbackoff true


#parameters below, need to be changed during the assignment
set exponinc false
set additiveinc true
set slowstart false
set fastrtx false

# create a TCP connection
set tcp1 [$nssim create-connection TCP/RFC793edu $na1 TCPSink $nb2 1]
$tcp1 set fid_ 1

#TCP-RFC793edu parameters

$tcp1 set rtxcur_init_ $initialtimeout
$tcp1 set add793jacobsonrtt_ $jacobsonrtt
 $tcp1 set add793karnrtt_ $karnrtt
 $tcp1 set add793rto_ $rto
 $tcp1 set add793syn_ $syn
 $tcp1 set add793expbackoff_ $expbackoff
 $tcp1 set add793fastrtx_ $fastrtx
 $tcp1 set add793slowstart_ $slowstart
 $tcp1 set add793additiveinc_ $additiveinc
 $tcp1 set add793exponinc_ $exponinc
 $tcp1 set packetSize_ $mpktsize

# The command below sets the slow start threshold to 1 at the
# beginning of the simulation, uncomment when necessary

#$nssim at 0.0 "$tcp1 set ssthresh_ 1"
```

With the TCP connection established, we need to create the ftp application and attach it to the TCP source

```
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
```

And then we set some parameter values

```
$tcp1 set packetSize_ $mpktsize
$tcp1 set window_ $maxwnd
```

UDP agent

For the background traffic we create the UDP connection in the same way, but instead of using a given application, we create the procedure `sendUDP` to send UDP packets at exponentially distributed intervals.

The UDP source at node na2.

```
set udp1 [new Agent/UDP]
$nssim attach-agent $na2 $udp1
```

The UDP sink at node nb2 is a null sink, as there is no need to send acknowledgments

```
set null1 [new Agent/Null]
$nssim attach-agent $nb2 $null1
```

And we connect the two agents

```
$nssim connect $udp1 $null1
```

The packet size of the UDP flows set to

```
$udp1 set packetSize_ $mpktsize
```

and the flow ID to default value of 0

```
$udp1 set fid_ 0
```

Then we define a procedure to fill the bottleneck link with UDP flows, so that the load is 0.6. We define global parameters, set the time, send an UDP flow, and generate the next time of invoking `sendUDP`.

```
proc sendUDP {} {
    global udp1 mpktsize nssim rng bnbw
    set now [$nssim now]
    $udp1 send $mpktsize
    set exprnd [$rng exponential [expr $mpktsize*8/($bnbw*0.6)]]
    $nssim at [expr $now+$exprnd] "sendUDP"
}
```

For the exponential intervals we use a random number generator, which we need to create. We set the seed to always be the same, in order to compare the different scenarios. With command `$rng seed 0` the seed rng is seeded heuristically.

```
set rng [new RNG]
$rng seed predef 1
```

We then schedule the sending of TCP and UDP packets at the beginning of the simulation

```
$nssim at 0 "sendUDP"
$nssim at 0 "$ftp1 start"
```

Monitoring and gathering data

Tracing In order to study the window evolution of the TCP connection, we need to open the file `win.dat` for writing and trace the value of the congestion window for our source `tcp1`.

When using the RFC793edu TCP agent we need to make changes to the tracing of the congestion window. We cannot use the variable `cwnd_` as given by the agent, but need to take into account the maximum advertised window. The procedure is

```
set winfile [open win.dat w]

proc record {} {
    global nssim tcp1 winfile maxwnd

    #Sampling interval
    set time 0.01

    #WINDOW FOR TCP1
    #Read congestion window of tcp1
    set curr_cwnd [$tcp1 set cwnd_]

    #Compare the value of cwnd to maxwnd to decide actual sending window
    if { $curr_cwnd < $maxwnd } {
        set curr_wnd $curr_cwnd
    } else {
        set curr_wnd $maxwnd
    }
    #current time
    set now [$nssim now]

    # write the file as window vs time trace
    puts $winfile "$now $curr_wnd"

    #re-schedule procedure
    $nssim at [expr $now+$time] "record"
    }
```

Before the simulation is ended the file has to be closed.

```
close $winfile
```

Monitoring Instead of tracing all events from the simulation we will monitor some relevant statistics of our simulation.

Let us monitor the flows. We set up the flow monitor between the bottleneck link, that is between nodes nb1 and nb2 with the following commands (note that `attach-fmon` is defined in the simulator class `$nssim`)

```
set fmon [$nssim makeflowmon Fid]
$nssim attach-fmon [$nssim link $nb1 $nb2] $fmon
```

The `fmon` monitors all the flows, but we are also interested in monitoring the individual flows. Then we can create a procedure to track the statistics of the flows. First we set

```
set fclassifier [$fmon classifier]
set flow1 [$fclassifier lookup auto 0 0 1]
```

and then we gather the information we need for that flow(`flow1`) and for the link in question (`fmon`), namely packet arrivals and packet drops.

```
set parrtcp [$flow1 set parrivals_]
set pdropstcp [$flow1 set pdrops_]
puts "TCP: $parrtcp $pdropstcp"
set parr [$fmon set parrivals_]
set pdrops [$fmon set pdrops_]
```

The monitoring part, where we monitor statistics on the bottleneck link for the TCP and UDP flow thus looks like

```
#set monitoring

set fmon [$nssim makeflowmon Fid]
$nssim attach-fmon [$nssim link $nb1 $nb2] $fmon

proc flowstats {} {
    global fmon
    set fclassifier [$fmon classifier]
    set flow1 [$fclassifier lookup auto 0 0 1]
    set flow2 [$fclassifier lookup auto 0 0 0]
    set parrtcp [$flow1 set parrivals_]
    set pdropstcp [$flow1 set pdrops_]
    puts "TCP: $parrtcp $pdropstcp"
    set parrudp [$flow2 set parrivals_]
    set pdropsudp [$flow2 set pdrops_]
    puts "UDP: $parrudp $pdropsudp"
    set parr [$fmon set parrivals_]
    set pdrops [$fmon set pdrops_]
    puts "All: $parr $pdrops"
}

set winfile [open win.dat w]
proc record {} {
    global nssim tcp1 winfile maxwnd

    #Sampling interval
    set time 0.01

    #WINDOW FOR TCP1
    #Read congestion window of tcp1
    set curr_cwnd [$tcp1 set cwnd_]

    #Compare the value of cwnd to maxwnd to decide actual sending window
    if { $curr_cwnd < $maxwnd } {
        set curr_wnd $curr_cwnd
    } else {
        set curr_wnd $maxwnd
    }
    #current time
```

```
set now [$nssim now]

# write the file as window vs time trace
puts $winfile "$now $curr_wnd"

#re-schedule procedure
$nssim at [expr $now+$time] "record"
}
```

With the procedure record scheduled at the beginning and the procedure flowstats scheduled at the end of the simulation

Several TCP sources

In question 2 we do not use UDP to fill the bottleneck link, but have many TCP sources. The code creates altogether 12 TCP connections, six to both of the two access links. All the connections in one access link have the same TCP agent.

To create many TCP sources a for-loop has been used. For example (TCP Tahoe sources from one access link)

```
for {set i 1} {$i < $imax} {incr i} {
    set tcp($i) [$nssim create-connection TCP $na1 TCPSink $nb2 $i]
    set ftp($i) [new Application/FTP]
$ftp($i) attach-agent $tcp($i)
$tcp($i) set packetSize_ $mpktsize
$tcp($i) set window_ $maxwnd
}
```

Note that you now have a variable `$tcp`, which is an array, whose ith element is `$tcp($i)`.

Then when monitoring all TCP sources a procedure has been used that returns all necessary information. You can create monitors to study other aspects of the simulation, but the information that the ns simulation automatically returns is enough to answer the exercise.

Running the simulation

```
$nssim at 0 "sendUDP"
$nssim at 0 "$ftp1 start"
$nssim at 0 "record"
$nssim at $simtime "flowstats"
$nssim at $simtime "halt"
```

Processing window data Before ending our simulation, we can pre process the data collected from tracing the congestion window.

```
# process window data
exec awk {
    {if ($1 != "") print $1, $2}
} win.dat > winfinal.dat
exec rm win.dat
```

Ending exit 0

2 Hints on Questions

For the first question use the example code question1.tcl and for the second question use the example code question2.tcl available on the course webpage. Make only the necessary changes to the codes.

To obtain the required options in question 1:

I) Use RFC793edu TCP agent with additive increase and multiplicative decrease

- set additiveinc true, others false, set the slow start threshold to 1

II) Use RFC793edu TCP agent with exponential increase and multiplicative decrease

- set exponinc true, others false, set the slow start threshold to 1

III) Use RFC793edu TCP agent with additive increase and slow start

- set slowstart true, others false

IV) Use RFC793edu TCP agent with additive increase, slow start and fast retransmission

- set slowstart true, set fastrtx true, others false

V) Use the TCP/Reno agent.

VI) Use the TCP/Vegas agent.

• To set the slow start threshold to 1 uncomment the line

```
$nssim at 0.0 "$tcp1 set ssthresh_ 1"
```

Otherwise keep it commented.

• In V and VI as well as question 2, you need to use other TCP agents than just the TCP/RFC793edu agent. To do this, you need to make the changes in the code to the line

```
set tcp($i) [$nssim create-connection TCP/RFC793edu $na1 TCPSink $nb2 $i]
```

command	TCP version
TCP/RFC793edu	TCP RFC793edu
TCP	TCP Tahoe
TCP/Reno	TCP Reno
TCP/Vegas	TCP Vegas

NOTE! When using agents other than TCP/RFC793edu, comment the lines under

```
#TCP-RFC793edu parameters
```

How to plot the congestion window evolution

- You can use any software you like, but one good choice is Matlab
- Matlab is started with commands *use matlab* and then *matlab*
- Exit Matlab by command *exit*
- The congestion window can be plotted in matlab by the following commands:

```
load -ascii winfinal.dat
time=winfinal(:,1)
window=winfinal(:,2)
plot(time,window)
```