Guidelines for writing and executing tcl scripts:

NS2 -- Network simulator simulates the TCL scripts. Hence every file that you write are in form "<filename>.tcl".

You can use the "vi" editor in the cygwin / linux systems for editing the files.

NS2 Power point presentation will help in understanding the scenario.

You can also use to NSCRIPT to generate these files. CHECK the links section

Here follows a sample TCL script :

Please note :: THERE ARE 3 SAMPLES :: PLEASE READ CAREFULLY & TRY to UNDERSTAND

"#" in TCL script refers to comments. Unless or otherwise specified all the entries are necessary.
****

Sample 1:

```
# Setting the Simulator start point
set ns [new Simulator]

#Define different colors for data flows (for NAM)-- More colors can be added in the same fashion
#this color no. corresponds to flow id of TCP/UDP connection
$ns color 1 Blue
$ns color 2 Red
        ...
        ...
        ...

#Open the Trace file -- "<filename>" refers to trace file name
set file1 [open <filename>.tr w]
$ns trace-all $file1

#Open the NAM trace file -- "<filename1>" refers to NAM file Network Animator
set file2 [open <filename1>.nam w]

#for WINDOW size tracing only
set winfile [open <FileName> w]
$ns namtrace-all $file2
```

```
#Define a 'finish' procedure
proc finish {} {
        global ns file1 file2
        $ns flush-trace
        close $file1
        close $file2
        exec nam <filename1>.nam &
        exit 0
}

# Needed only for Dynamic routing i.e. Assignments 3,4
$ns rtproto DV


#Creating nodes -- The following commands will generate 2 nodes ,
#for more nodes more statements can be added in same fashion
set n0 [$ns node]
set n1 [$ns node]
            ...
            ...
            ...

#Create links between the nodes -- 0.3Mb is BW; 10ms is delay ; DropTail is buffer type
#"duplex-link" corresponds to duplex link between nodes 0 & 1
$ns duplex-link $n0 $n1 0.3Mb 10ms DropTail
            ...
            ...
            ...
#only for simplex links -- Please NOTE you have two statements for simplex
$ns simplex-link $n0 $n1 0.3Mb 10ms DropTail
$ns simplex-link $n1 $n0 0.3Mb 10ms DropTail


#setting up LAN ONLY -- implies nodes 3,4,5 together as LAN with 0.5Mb BW 40ms
delay
#LL refers to LinkLayer 802_3 refers to 802.3 protocol
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/802_3
Channel]


#Give node position (for NAM) -- just needed for orientation try this urself
$ns duplex-link-op  $n0 $n1 orient right
            ....
            ....
            ....
            ....
```

....

```
#Setup a TCP connection -- these two lines define a TCP source agent over node 0
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp

# setting up TCP receiver -- these lines define a TCP sink or receiver over node 4
# these set of lines with above lines imply connection between node 0 & node 4
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink

# specifies flow color means Blue defined earlier
$tcp set fid_ 1


# setting Qsize required in some exercises
#implies Q size of link connecting 4&5 is 10
$ns queue-limit $n4 $n5 10

#for random drops to set random drops required in only some assignements
#for random drops between nodes x & y
set loss_module [new ErrorModel]
$loss_module set rate_ 0.0
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $nx $ny



#Setting up a UDP connection -- if required
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $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
```

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

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

# for dynamic routing only these statements implies n1-->n4 link is down & up
$ns rtmodel-at 0.3 down $n1 $n4
$ns rtmodel-at 1.5 up $n1 $n4

# for plotting TCP window size in certain questions only
# Window size w.r.t time
proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"

# for starting simulations w.r.t ftp ; if CBR change the same
# implies the start time
$ns at 0.1 "$ftp start"
$ns at 0.5 "$ftp1 start"

# ending simulations
$ns at 6.0 "finish"

$ns run
```

****

Second sample w.r.t. multicasting :

****

```
# Setting the Simulator start point
```

```
set ns [new Simulator]

# for multicast applications only
$ns multicast

#Define different colors for data flows (for NAM)-- More colors can be added in the
same fashion
#this color no. corresponds to flow id of TCP/UDP connection  -- use the above example

# Generate trace files -- use the above example


# allocate a multicast address for a group for multicast applications;
set group [Node allocaddr]

# nod is the number of nodes setting a variable nod
set nod 6

# use finish procedure use from first example

# using arrays to create multicast capable nodes  ;
for {set i 1} {$i <= $nod} {incr i} {
  set n($i) [$ns node]
}

#allocating duplex-links for the nodes created above since arrays use indexing
$ns duplex-link $n(1) $n(2) 0.3Mb 10ms DropTail

                    ....
                    ....
                    ....
                    ....
                    ....



#align nodes :: use the above technique along with format of first example
                    ....
                    ....
                    ....
                    ....
                    ....
```

```
# configure multicast protocol;
set mproto CtrMcast

# all nodes will contain multicast protocol agents;
set mrthandle [$ns mrtproto $mproto]

# set RV and bootstrap points
$mrthandle set_c_rp $n(5)


# Set UDP / TCP agents with CBR/FTP traffic respectively as required -- use first
example
                ....
                ....
                ....
                ....
                ....


# create receiver agents -- needed for leaving and joining groups in multicasting
set rcvr [new Agent/LossMonitor]

# joining and leaving the group -- implies node 2 joins the group & leaves the group
# at specified times
$ns at 0.6 "$n(3) join-group $rcvr $group"

$ns at 1.9 "$n(3) leave-group $rcvr $group"

# specify  start times and end times - use first example.

# run ns command
$ns run
```

****

Sample 3 for Q monitoring::

```
# Setting the Simulator start point
set ns [new Simulator]

# Generate trace files -- use the above example

# set the parameters for lambda and mu
```

```
set lambda 30.0
set mu     33.0

# set nodes use first example

# use finish procedure use from first example

# set the link

set link [$ns simplex-link $n1 $n2 xkb yms <buffertype>]

# set qsize -- use first example

# for generating random arrival times and packet sizes
# usage of mu and lamda here

set InterArrivalTime [new RandomVariable/Exponential]
$InterArrivalTime set avg_ [expr 1/$lambda]
set pktsize [new RandomVariable/Exponential]
$pktsize set avg_ [expr <qsize>/(8*$mu)]


# set UDP agent source and destination-- use first example

# IMP :: Q monitoring ::
set qmon [$ns monitor-queue $n1 $n2 [open <filename>.out w] 0.1]
$link queue-timeout

# procedure for sending packets

proc sendpacket { } {

        global ns src InterArrivalTime pktsize
        set time [$ns now]
        $ns at [expr $time + [$InterArrivalTime value]] "send packet"
        set bytes [expr round ([$pktsize value])]
        $src send $ bytes

        }

# send packets using sendpacket{} procedure

$ns at 0.0001 "sendpacket"

# terminate the simulation -- use first example
```

```
# final statment
$ns run
```

****

For throughput calculation use the following PERL script::

name this file at "throughput.pl" :

****

```
$infile=$ARGV[0];
$tonode=$ARGV[1];
$granularity=$ARGV[2];

#we compute how many bytes were transmitted during time interval specified
#by granularity parameter in seconds
$sum=0;
$clock=0;

     open (DATA,"<$infile")
       || die "Can't open $infile $!";

   while (<DATA>) {
         @x = split(' ');

#column 1 is time
if ($x[1]-$clock <= $granularity)
{
#checking if the event corresponds to a reception
if ($x[0] eq 'r')
{
#checking if the destination corresponds to arg 1
if ($x[3] eq $tonode)
{
#checking if the packet type is TCP
if ($x[4] eq 'tcp')
{
   $sum=$sum+$x[5];
}
}
}
}
else
```

```
{   $throughput=$sum/$granularity;
    print STDOUT "$x[1] $throughput\n";
    $clock=$clock+$granularity;
    $sum=0;
}
}
   $throughput=$sum/$granularity;
    print STDOUT "$x[1] $throughput\n";
    $clock=$clock+$granularity;
    $sum=0;

    close DATA;
exit(0);
```

****

For calculating throughput u need the following along with the above perl script::

trace file -- <filename>.tr ;

use the following command ::

> perl throughput.pl <trace file> <required node> <granlarity> > <output file>

for getting the graph of throughput ::

use gnuplot.

IMP LINKS::

1) Trace graph : http://www.geocities.com/tracegraph/
2) Gnuplot: http://www.gnuplot.info/
3) Nscript : http://home.gwu.edu/~ecamposn/software.html
4) NS homepage : http://www.isi.edu/nsnam/ns/