CSC645/745 COMPUTER NETWORKS

Project 2
Network Simulations Using NS2

1. Instruction

In a complex, dynamic, expensive, and large-scale system like the Internet, analytic solutions are usually not known or are only approximately known. Due to many practical reasons such as cost, risk, and controllability, etc., real experiments on the actual network system may not be preferred. Network simulators attempt to represent real world networks by modeling key elements of the real system and incorporating most of its salient features. Furthermore, it is not too complex for us to understand and experiment with them. The models and their operations enable us to predict the behavior of actual systems at low cost under different configurations of interest and over long period. Although network simulators are not perfect, they are usually accurate enough to give us a meaningful insight into how the network is working, and how its operation can be optimized.

Many network simulators, such as NS2, Openet, Qualnet, etc., are widely available. We'll use NS2 for this project. NS2 is a discrete event simulator written in C++, with an OTcl interpreter shell as the user interface that allows the input model files (Tcl scripts) to be executed. Most network elements in the NS2 simulator are developed as classes, in object-oriented fashion. The simulator supports a class hierarchy in C++, and a very similar class hierarchy in OTcl. The root of this class hierarchy is the TclObject in OTcl. Users create new simulator objects through the OTcl interpreter, and then these objects are mirrored by corresponding objects in the class hierarchy in C++. NS2 provides substantial support for simulation of TCP, routing algorithms, queueing algorithms, and multicast protocols over wired and wireless (local and satellite) networks, etc. It is freely distributed, and all source code is available.

Developing new networking protocols and creating simulation scripts are complex tasks, which requires understanding of the NS2 class hierarchy, C++, and Tcl programming. However, in this project, you only need to design and run simulations in Tcl scripts using the simulator objects without changing NS2 core components such as the class hierarchy, event schedulers, and other network building blocks. This project has to be done individually but not in groups.

2. Goal

- Familiarize yourself with network simulations using NS2, which is a widely used network analysis tool.
- Learn all the basic steps to perform a complete NS2 simulation.
- Study the operation and effects of different window sizes in TCP

3. NS2 Installation

Below are the steps to download and install NS2 and related software packages on Ubuntu 14.04. NS2 also has a Windows version and can be installed on Windows. However, it doesn't work as well as the UNIX/LINUX version.

Step 0: Install Ubuntu 14.04
Ubuntu 14.04 (32-bit) can be downloaded here http://www.ubuntu.com/download/alternative-downloads
If you use a Mac or Windows computer, you can run Ubuntu on Virtualbox. Here is a tutorial video on how to install Ubuntu 14.04 on Virtualbox.
https://www.youtube.com/watch?v=kz7TcHW2UTc

Step 1: Download NS2
Download all-in-one NS2 package version 2.35 from the NS2 web site
https://sourceforge.net/projects/nsnam/files/allinone/ns-allinone-2.35/ns-allinone-2.35.tar.gz/download
The package downloaded will be named "ns-allinone-2.35.tar.gz". Copy it to the home folder. In a terminal, use the following command to extract the contents of the package:

```
tar zxvf ns-allinone-2.35.tar.gz
```

All the files will be extracted into a folder called "ns-allinone-2.35".

Step 2: Install Related Software Packages
Use the following commends to install some related software packages.

```
sudo apt-get update
sudo apt-get install build-essential autoconf automake libxmu-dev
```

Step 3. Modify "ls.h" File
Use the following commend to open "ls.h" file.

```
sudo gedit ns-2.35/linkstate/ls.h
```

Change the 137th line from

```
void eraseAll() { erase(baseMap::begin(), baseMap::end()); }
```

to

```
void eraseAll() { this->erase(baseMap::begin(), baseMap::end()); }
```

Step 4. Install NS2
Use the following command to install NS2.

```
./install
```

Step 6. Set the Environment Path
Before we run NS2, we need to add the build path to the environment path. Use the following command to open the ".bashrc" file.

```
cd
sudo gedit .bashrc
```

Add the following lines to the bottom of the file. Change the "UserName" in red to your username in Ubuntu.

```
export PATH="$PATH:/home/UserName/ns-allinone-2.35/bin:/home/UserName/ns-
allinone-2.35/tcl8.5.10/unix:/home/UserName/ns-allinone-
2.35/tk8.5.10/unix"
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/home/UserName/ns-allinone-
2.35/otcl-1.14, /home/UserName/ns-allinone-2.35/lib"
export TCL_LIBRARY="$TCL_LIBRARY:/home/UserName/ns-allinone-
2.35/tcl8.5.10/library"
```

Once the changes has been made, save the file and restart the system. (Or use the command `source /home/UserName/.bashrc` to source the path we add.)

Step 7. Run NS2
Open a terminal and start NS2 by using the following command:

```
ns
```

If you see "%", it means the installation is successful. Then, type "exit" to exit.

4. How to Use NS2

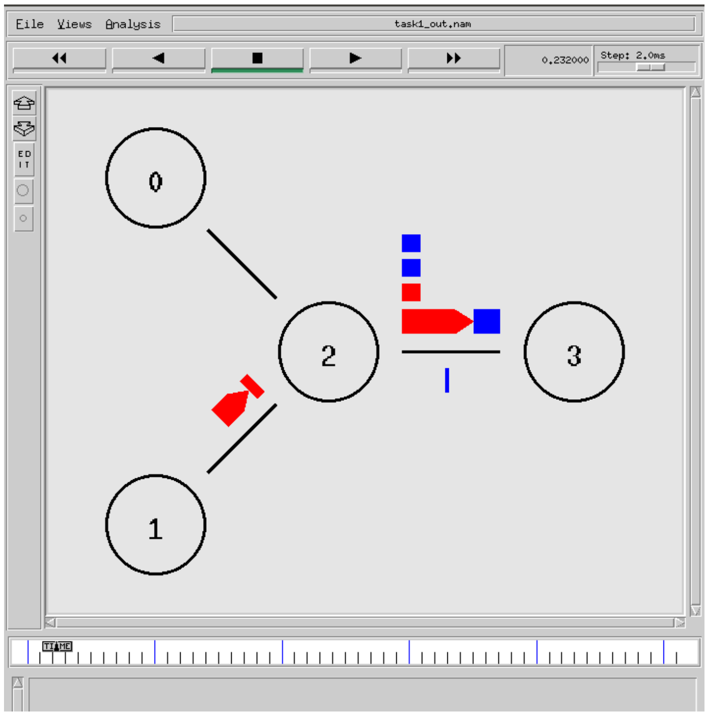
In this project, you need to run simulations in Tcl scripts in NS2. You can download a sample simulation file named "task1_sample.tcl" from iLearn. In order to run it, Enter to the directory that stores your simulation files and use the following command

```
ns task1_sample.tcl
```

The execution of "task1_sample.tcl" produces a trace file "task1_out.nam". You can use "nam" program to visualize the entire simulation by typing following command

```
nam task1_out.nam
```

The GUI interface is shown below. You can press the play button to begin the animation.



5. Task: Bandwidth Share of TCP flows with different Window Sizes

Consider a network with 4 nodes, as shown in Figure 1. The link between node 3 and node 4 has a speed of 1 Mbps. All other links have a speed of 2 Mbps. All links have a propagation delay of 10 ms and a Drop Tail buffer. There is a TCP/FTP flow from node 1 to node 4 and from node 2 to node 4. Both start at 0.1 second, and end at 5 seconds. The packet length for both flows is 1000 Bytes, and they both have a maximum window size of 1 packet. The queue buffer size for the link from node 3 to node 4 is 4 packets. There is no other traffic in the network other than that described above.

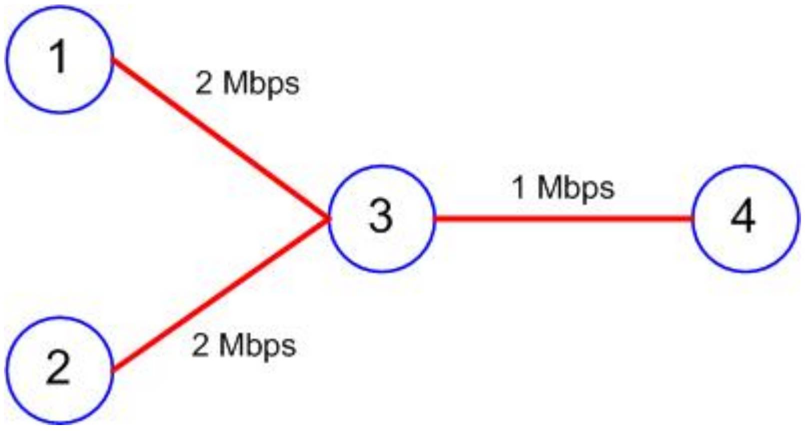


Figure 1

You are given a sample file (task1_sample.tcl) to simulate the above-mentioned network. Run the simulation for 5 seconds by typing "ns task1_sample.tcl", and then we obtain an output trace file, task1_out.nam. You need to understand the trace format and be able to extract meaningful performance metrics from such files by using Awk. Take a look at "task1_out.nam". Ignore entries with -t *; these entries are used by nam to visualize the network topology. The rest of the entries look like:

- [event type] -t [time] -s [src node] -d [dst node] -p [pkt type] -e [pkt size] -c [color] -i [pkt id] -a [flow id] -x {[src.port] [dst.port] [seqno] ------- null}

The fields in the trace file are: type of the event, simulation time when the event occurred, source and destination nodes, packet type (protocol, action or traffic source), packet size, color type, packet ID, flow identifier, source and destination addresses, sequence number, and packet id. Note that the internal identifier numbers for node 1, node 2, node 3, and node 4 are 0, 1, 2, and 3, respectively. The event type "+", "-", "r", and "d" represent enqueue operations, dequeue operations, receive events, and drop events, respectively.

- If we want to extract information from the "task1_out.nam" file and compute the total bytes of the first TCP/FTP flow over the link from node 3 to node 4, we run the following Awk command:

```
awk '$1== "r" && $7==3 && $17==1 && $9=="tcp" {a += $11} END {print
a}' task1_out.nam
```

In this command, "$1", "$7", "$17", and "$9" mean the 1st (event type), 7th (destination node), 17th (flow identifier), and 9th (packet type) columns of each line in file "task1_out.nam". "a" is an Awk variable that is used to calculate the total bytes. Awk scans the input file "task1_out.nam" line by line, and parses the line column by column.

- Similarly, you can run the following Awk command to compute total bytes of the second TCP/FTP flow:

  ```
  awk '$1=="r" && $7==3 && $17==2 && $9=="tcp" {b += $11} END {print b}'
  task1_out.nam
  ```

- Dropped packets are measured simply with:

  ```
  awk '$1=="d" {c += $11} END {print c}' task1_out.nam
  ```

- Now you can compute the total bytes of TCP/FTP traffic and the ratios of such traffic by using a more elaborate Awk script:

  ```
  awk '{if ($1== "r" && $7==3 && $17==1 && $9=="tcp") {a += $11} else if
  ($1== "r"  && $7==3 && $17==2 && $9=="tcp") {b += $11} else if
  ($1=="d") {c += $11}} END {printf (" tcp1:\t%d\n tcp2:\t%d \n
  total:\t%d\n tcp1 ratio:\t%.4f \n tcp2 ratio:\t%.4f\n Dropped
  Packets:\t%d\n ", a, b, a+b, a/(a+b), b/(a+b), c)}' task1_out.nam
  ```

Next, let's take a look at the Tcl scripts in "task1_sample.tcl", and go through the sequence of required steps to set up and solve a simulation problem:

- As the first step, we create the simulator object.

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

- Set up the network topology by creating node objects, and connecting the nodes with link objects. If the output queue of a router is implemented as a part of a link, we need to specify the queue type. In this project, DropTail queue is used. In some cases, we may define the layout of the network topology for better NAM display.

  ```
        # create four nodes
        set node1 [$ns node]
        set node2 [$ns node]
        set node3 [$ns node]
        set node4 [$ns node]

        # create links between the nodes
        $ns duplex-link $node1 $node3 2Mb 10ms DropTail
        $ns duplex-link $node2 $node3 2Mb 10ms DropTail
  ```

```
$ns duplex-link $node3 $node4 1Mb 10ms DropTail
$ns queue-limit $node3 $node4 4

# set the display layout of nodes and links for nam
$ns duplex-link-op $node1 $node3 orient right-down
$ns duplex-link-op $node2 $node3 orient right-up
$ns duplex-link-op $node3 $node4 orient right

# define different colors for nam data flows
$ns color 0 Green
$ns color 1 Blue
$ns color 2 Red
$ns color 3 Yellow

# define the position of queue between node3 and node4 in nam
$ns duplex-link-op $node3 $node4 queuePos 0.5
```

- Define traffic patterns by creating agents, applications and flows. In NS2, packets are always sent from one agent to another agent or a group of agents. In addition, we need to associate these agents with nodes.

```
# First TCP traffic source
# create a TCP agent and attach it to node node1
set tcp1 [new Agent/TCP]
$ns attach-agent $node1 $tcp1
$tcp1 set fid_ 1
$tcp1 set class_ 1

# window_ * (packetsize_ + 40) / RTT
$tcp1 set window_ 1
$tcp1 set packetSize_ $packetSize

# create a TCP sink agent and attach it to node node4
set sink1 [new Agent/TCPSink]
$ns attach-agent $node4 $sink

# connect both agents
$ns connect $tcp1 $sink

# create an FTP source "application";
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1



# Second TCP traffic source
# create a TCP agent and attach it to node node1
set tcp2 [new Agent/TCP]
$ns attach-agent $node2 $tcp2
```

```
$tcp2 set fid_ 2
$tcp2 set class_ 2

# window_ * (packetsize_ + 40) / RTT
$tcp2 set window_ 1
$tcp2 set packetSize_ $packetSize

# create a TCP sink agent and attach it to node node4
set sink2 [new Agent/TCPSink]
$ns attach-agent $node4 $sink2

# connect both agents
$ns connect $tcp2 $sink2

# create an FTP source "application";
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
```

- Define the trace files, and place monitors at places in the topology to collect information about packets flows. NS2 supports two primary monitoring capabilities: traces and monitors. The traces enable recording of packets whenever an event such as packet drop or arrival occurs in a queue or a link. The monitors provide a means for collecting quantities, such as number of packet drops or number of arrived packets in the queue. The monitor can be used to collect these quantities for all packets or just for a specified flow (a flow monitor).

```
# open the nam trace file
set nam_trace_fd [open taks1_out.nam w]
$ns namtrace-all $nam_trace_fd

#Define a 'finish' procedure
proc finish {} {
        global ns nam_trace_fd trace_fd

        # close the nam trace file
        $ns flush-trace
        close $nam_trace_fd

        # execute nam on the trace file
        exit 0
}
```

- Schedule the simulation by defining the start and stop of the simulation, traffic flows, tracing, and other events.

```
# schedule events for all the flows
$ns at 0.1 "$ftp1 start"
$ns at 0.1 "$ftp2 start"
```

```
$ns at 5.0 "$ftp2 stop"
$ns at 5.0 "$ftp1 stop"

# call the finish procedure after 6 seconds of simulation time
$ns at 6 "finish"

# run the simulation
$ns run
```

- Finally we need to extract useful data from the traces, as you just did using Awk commands, and feed it to plotting software to get human readable results.

(a) Based on the execution results of the Awk commands, what is the total traffic of each flows? What is the overall packet loss rate of the two TCP flows? What is the ratio of throughput between the two TCP flows?

(b) Revise the sample code "task1_sample.tcl" so that the window size of the second TCP flow (tcp2) changes from 1 to 25, and save it a new file “task1_b.tcl”. Use the Awk commands to compute the total traffic of each flow. How does the packet loss rate change? Which flow uses more bandwidth in the bottleneck from node 3 to node 4? What is the reason?

(c) Revise the sample code "task1_sample.tcl" so that the window size of both the TCP flows are set to 5, and save it as a new file “task1_c.tcl”. Use the Awk commands to compute the total traffic of each flows. How does the packet loss rate change? Do both flows receive a "fair share" of the available bandwidth of the bottleneck link?

(d) Revise the sample code "task1_sample.tcl" so that the window size of both the TCP flows are set to 5 and the queue buffer size for the link from node 3 to node 4 is set to 40 packets. Save the new file as “task1_d.tcl”. Use the Awk commands to compute the total traffic of each flows. How does the packet loss rate change? Do both flows receive a "fair share" of the available bandwidth of the bottleneck link?

(e) Revise the sample code “task1_sample.tcl” and add node 5 to the network configuration. It is connected to node 3. The link between node 5 and node 3 has a speed of 2 Mbps and a propagation delay of 10 ms. The new network topology is shown in Figure 2. One more TCP/FTP flow with a window size of 5 is generated from node 5 to node 4. Save the new file as “task1_e.tcl”. Run the simulation. Revise the sample Awk commands to compute the total bytes of TCP for each flow. What are the Awk commands? Do all the TCP flows receive a fair "share" of the bandwidth? Why or why not?

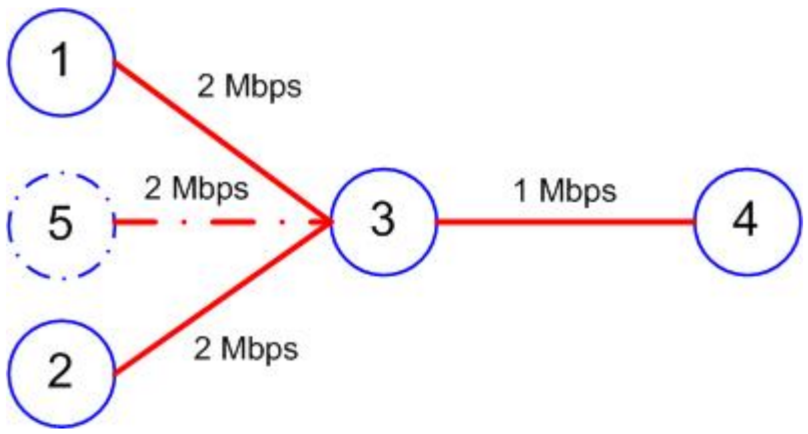


Figure 2

6. Submission

Write a report to answer ALL the questions in (a)-(e). Save it as "task1.pdf". Submit one ZIP file consisting of task1.pdf, task1_b.tcl, task1_c.tcl, task1_d.tcl, and task1_e.tcl to the regular submission link on iLearn.