# S-38.180: Quality of Service in Internet

## *Exercise 1: Packet latency and throughput*

**Deadline:** **29.10.2004, 4 pm**
**Grading:** **Pass / Fail**
**Return:** **PDF by e-mail (tviipuri@netlab.hut.fi) OR paper version to the course locker (in the 2nd floor, G-wing)**

### *Introduction*

Packet latency caused by the network has a significant effect on the throughput of a TCP-connection. TCP-protocol relies on the acknowledgements sent by the client-side of the connection in deciding the optimal send rate which is controlled by altering the size of the congestion window. The congestion window represents the maximum number of unacknowledged packets sent to the network. Initially the window has a low value but during the communication it is increased or decreased depending on the arrivals or lack of arrivals of acknowledgement packets. The more it takes for an acknowledgement to arrive the more time it takes for the send rate to be increased. In this exercise you will measure the effect of packet latency on the throughput of multiple TCP-connections all competing of the same network resources.

### *Network*

The topology of the network to be simulated is shown in illustration 1. The link connecting nodes 0 and 1 has a fixed bandwidth of 5 Mbps and packet delay of 20 ms. The links connecting nodes 2—4 to node 1 have the same bandwidth of 10 Mbps and their delays are to be altered in the simulations.

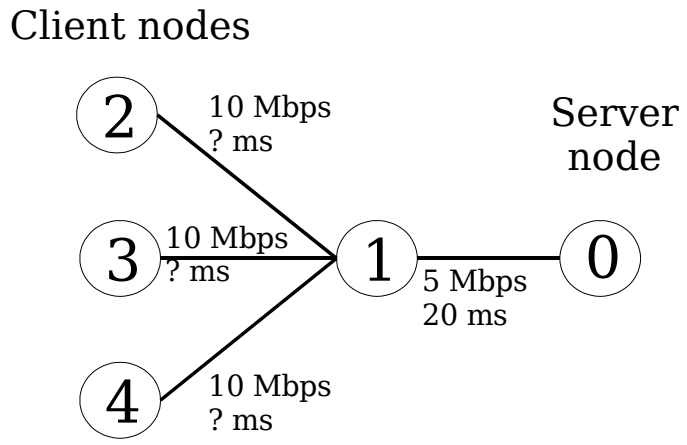


Illustration 1 Topology of the simulated network

## *Traffic Generation*

The traffic in the network will be created by defining three TCP-connections controlled by FTP-application. The corresponding classes in NS-2 are:

- Agent/TCP/FullTcp
- Application/FTP

The connection pairs are as follows:

1. server node 0 <-> client node 2
2. server node 0 <-> client node 3
3. server node 0 <-> client node 4

All connections use the link between node 0 and 1 and it will be the bottle-neck link in the network. The direction of the data transmission should be from the client nodes to the server node, i.e. the clients are uploading data to the server.

**Hint!** You should create 3 client and 3 server TCP-agents and attach them to the appropriate nodes. Also you should create 3 FTP-applications and attach them to the client TCP-agents. See NS-2 manual (PDF-version, section *"35.4 Simulated applications: Telnet and FTP*") for more information on how to use the FTP-application.

## *Queueing*

Define RED-queueing (Random Early Detection) in the link between nodes 0 and 1 (bottle-neck). Packet drops will occur only in the bottle-neck link in this network, so the queueing discipline of the other links is not relevant. The easiest way is to define the other links as DropTail-links.

Use the following code to create the the link and the RED-queue between nodes *n0* and *n1* and to set their parameters:

```
$ns duplex-link $n0 $n1 5Mb 20ms RED
set bn_queue [[$ns link $n1 $n0] queue]
$bn_queue set limit_ 35          ;# Queue length (packets)
$bn_queue set thresh_ 8          ;# Lower RED threshold (packets)
$bn_queue set maxthresh_ 24      ;# Upper RED threshold (packets)
$bn_queue set q_weight_ 0.3      ;# Weight-parameter used in the queue size calculation
$bn_queue set linterm_ 10        ;# Inverse of the drop probability in maxthresh_
```

## *Monitoring*

Collecting the simulation results is possible by tracing all the links or just the bottle-neck link. Traces can then be post-processed by using tools like 'awk' or 'grep' to get the needed information. However, flow monitoring is a much more efficient method for collecting traffic statistics from a single link – in this case the bottle-neck link. Flow monitors are used to collect information about TCP or UDP flows, such as the number of transmitted and dropped packets. The rules by which flows are separated can be defined by the user. The simplest way is to set a unique flow-ID on each flow and configure the flow monitor to use them. The following is an example of attaching a flow-monitor on a link:

```
# Create a flow monitor which uses flow IDs to separate different flows
set flow_mon [$ns makeflowmon Fid]

# Attach the flow monitor to the observe the link between nodes n1 and n0
$ns attach-fmon [$ns link $n1 $n0] $flow_mon 0

# Open a file “output_file.fmon” and set it as the output file of the flow monitor
$flow_mon attach [open output_file.fmon w]
```

Assign a flow ID for each TCP agent on the sending side. For example:

```
$clnt_tcp1 set fid_ 1
$clnt_tcp2 set fid_ 2
$clnt_tcp3 set fis_ 3
```

The monitor is called each time the statistics need to be printed on the output file. For example writing the statistics after 100 seconds of simulation is done with command:

```
$ns at 100 "$flow_mon dump"
```

## *Task*

Create the simulation as described above. Simulate 10 different scenarios where the delay on the three client side links is altered. Use a simulation time of 100 seconds in each scenario.

Measure the throughput received by the TCP-connections in each scenario. Present your results and explain how much packet latency affects the throughput.

**Hint!** In the first scenario, use the same initial delay of 10 ms in all links. For the remaining scenarios, modify the delays in two of the links while keeping the delay in one of the links always the same. Use a high enough scale in the delays (1 – 200 ms) so that the difference in the throughput results can be clearly seen.

## *Additional Information*

1) To get NS-2 working, first type:

*source /p/edu/s-38.180/usens2.csh*

You can then start a simulation by typing:

*ns example.tcl*

2) The first source of information is the NS-2 home page:

http://www.isi.edu/nsnam/ns/

Especially the manual should be useful. Use the PDF-version because it is more complete than the HTML-version.

3) Check the example script for hints on creating a simulation:

http://www.netlab.hut.fi/opetus/s38180/2004/exercises/H1/example.tcl