Network Simulation: Assignment

Nelson Matthys (Nelson.Matthys@cs.kuleuven.be)
Rafael Bachiller Soler (Rafael.Bachiller@cs.kuleuven.be)
Danny Hughes (Danny.Hughes@cs.kuleuven.be)

Computer Networks, 2014

1 Practicalities

1.1 Objective

The goal of the assignment is to acquire a better understanding of the TCP transport protocol. To do this, we will create TCP connections and study their behaviour when other network traffic is generated in our network infrastructure. Specifically, you will perform the following tasks:

- Investigate the behaviour of the TCP protocol. More specifically:
  - slow start
  - congestion avoidance
  - the congestion window
  - slow start threshold
- Create a better understanding how the TCP protocol behaves in real world infrastructures and scenarios:
  - We study KotNet/Telenet CATV cable modems and the effects capping traffic has on download/upload connections.
  - We investigate the behaviour of TCP Tahoe and TCP Reno when the network is loaded with realistic traffic such as Web traffic.

1.2 Evaluation criteria and deadline

You are allowed to work on this assignment in pairs of two students. You should hand in a printed report which answers the questions posed and provides the requested materials such as graphs or calculations. Make sure your graphs and calculations are scientifically correct. Finally, please include a printed out version of your TCL-code (this allows us to better interpret your graphs if needed).

The reports should be delivered via the white letterbox near the secretariat of the CS department before **noon 12 pm** on **Friday, May 2nd 2014**. Clearly indicate your name and student number, plus course title and teacher's name on the report, and make sure they are stapled (to make it easier for the secretariat).

The relationship between marks and study points are given in the following table:

Grade	Criteria
A	75-100
B	60-74
C	50-59
D	40-49
F	Below 40 or No Submission by Deadline

Table 1: Grading Criteria.

1.3 Hints

Firstly, during the next two sessions, you are allowed to work on the assignment, write the report, and invited to ask questions to the teaching assistants. Note that chapter 4 of the NS2 guide on Toledo contains useful code samples plus a summary of TCP behaviour.

Secondly, for your convenience, you can speed up analysis of trace data if you mechanise all post-processing. This can be done, for example, using scripts or by putting all commands directly inside a TCL procedure (e.g. `proc finish{} {...}`). Note you can supply a script as argument to `gnuplot` or `xgraph` containing the commands used for plotting.

Thirdly, Toledo contains a script to calculate the total throughput of a data stream with a given flow identifier (`fid_`). This allows to filter out traffic from irrelevant streams. Also, see page 24 in the "NS Simulator for beginners" for more information about the structure of trace files.

Finally, you are allowed to clarify individual numbers or points of interest on graphs using a marker pen.

2 Exercise 1: Bandwidth restrictions on KotNet

In this exercise, we simulate the behaviour of a KotNet/Telenet cable modem connection and investigate how bandwidth restrictions influence traffic.

2.1 Context

Figure 1 shows the infrastructure of such a connection. The local network (one or more PCs) is connected with the Ethernet interface of the cable modem. The cable modem is connected through a 75 Ohm CATV coax cable with the Telenet coax network. At the other side of the coax network, a head-end connects the CATV network to the KU LeuvenNet network infrastructure.

On this network infrastructure, several bandwidth restrictions are in place:

Telenet cable modem The Telenet cable modem is capped for both download and upload. Current limitations for an individual KotNet/Telenet user are as follows: (`https://admin.kuleuven.be/icts/services/kotnet/2012`):

- Download: 4 Mbps
- Upload: 256 Kbps

CATV cable segment Also the bandwidth between the head-end and the different cable modems on the CATV is limited. Some cable segments for example have a bandwidth of 100Mb/s.

KotNet userquota and bandwidth regulations Additionally, all KotNet computers are restricted by userquota (both per login and per machine) and a bandwidth policer. More information on those regulations can be found at `http://www.kulnet.kuleuven.ac.be/bcrouter/`.

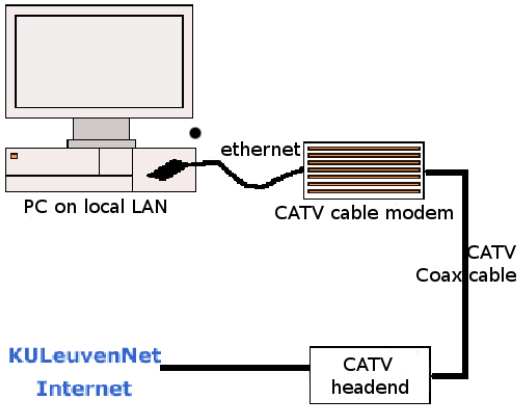

Figure 1: KotNet/Telenet infrastructure

In this exercise however, we only simulate the behaviour of one Telenet cable modem (of an individual KotNet/Telenet user). In this simplification we do not take into account the capacity of the CATV head-end and the Kotnet user quota and bandwidth regulations.

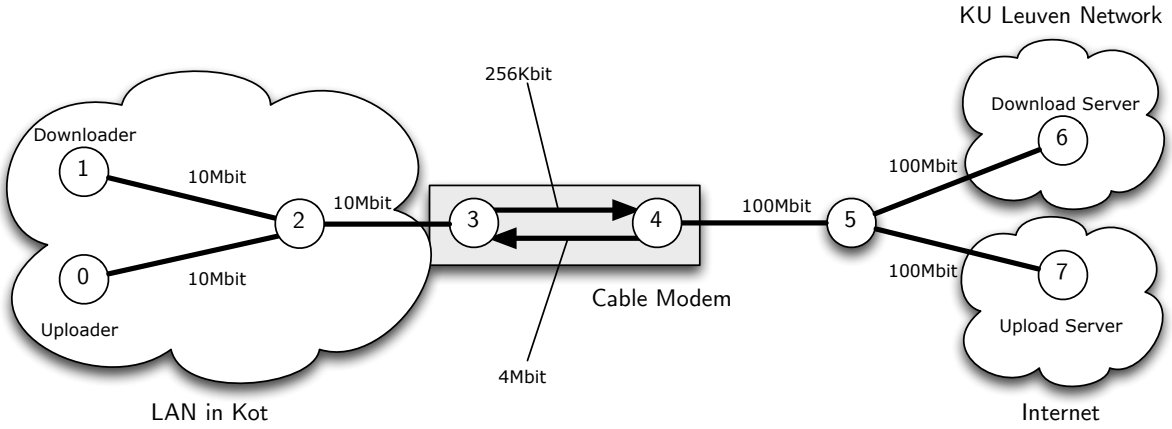

Figure 2: Simplified KotNet topology.

2.1.1 Topology

Figure 2 illustrates a simplified version of a typical KotNet setup. Node 0 and 1 are two computer nodes on the local LAN after the cable modem. They are both connected with a bidirectional link to a router or switch, which is on its turn connected to the cable modem. All links inside the LAN have a bandwidth of 10Mbps and a propagation delay of 0.2ms.

Node 3 and 4 represent the internals of the cable modem connection. We define the link between the modem and head-end by two *simplex links* with a propagation delay of 0.2ms. The link from 3 to 4 has a bandwidth of 256kbps, the other direction a bandwidth of 4Mbps. In fact, in this simulation we simplify the modem cap by simulating a slower, asymmetric link between the two nodes in the modem itself (node 3 and 4).

Link 4 to 5 represents the CATV cable segment and has a bandwidth of 100Mbps and a propagation delay of 0.3ms. After the cable segment, we have node 6 and 7 bidirectionally connected to the CATV head-end with a bandwidth of 100Mb/s and a propagation delay of 0.3ms.

2.1.2 Applications

We define two applications. The first application is a FTP application that represents a user who is downloading data from a server within the KU Leuven network. The second connection is a CBR connection, which represents a user that is uploading data to a server on the Internet. The FTP connection (obviously) uses an underlying TCP connection with a window (`window_`) value of 80 and default packetsize. The CBR connection uses UDP as transport protocol and a packetsize of 1500 bytes.

	FTP	CBR
From node	6	0
To node	1	7
Starting at	0.1 s	3.0 s
Stopping at	9.9 s	6.0 s

2.2 Questions (55p)

1. First, leave out the connection of the uploader. Plot the throughput of the main FTP connection and discuss the download behaviour of a KotNet/Telenet modem with bandwidth caps. How do bandwidth limitations affect the download connection?

2. What happens if you now enable both upload and download activities? Plot and compare the throughput of both streams. Do you spot any problems? If so, explain.

3. Consider now that the model is able to guarantee a certain amount of upload bandwidth to every application, which performance do you expect for both applications (FTP and CBR)?

4. What would be the result if bandwidth was even more restricted? For example, up to September 2010, KotNet limitations were 512Kbps for downstream traffic and 100Kbps for upstream traffic.

5. Configure the upload connection with a fixed `rate_` of 30k. Explain the simulated result (no plot required). What would be your (extra) solution(s) to improve the download experience on shared LANs behind capped Internet connections? How could you practically realise this?

6. Suppose we have in the future a KotNet/Telenet cable subscription that has 10 times more capacity: i.e a connection that has a downstream capacity of 40Mbps with an upstream capacity of 2Mbps.

    1. What performance can you expect if there are 10 users behind this connection and all performing the same activities as in this exercise? Why? What performance can you expect if there are only 5 users using the network? Why?

    2. Does the performance differ if 10 users perform the same activities but at random times? Are there some users who suffer from the network activities caused by other users? Why?

3 Exercise 2: Tahoe and Reno versus bursty web traffic

The goal of this exercise is to investigate TCP behaviour when the network is loaded with realistic traffic. In this exercise we consider traffic that is generated by users browsing web pages on the Internet.

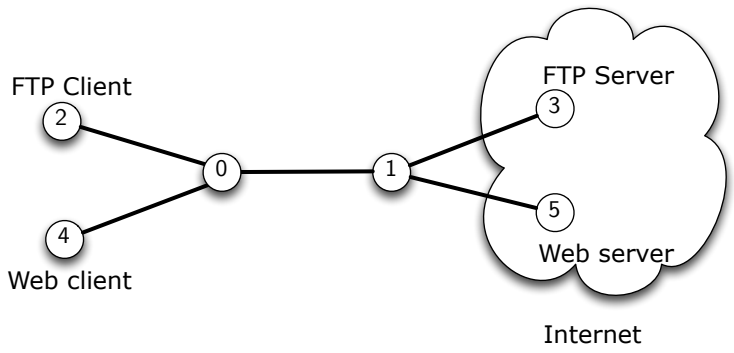

Figure 3: Topology of exercise 2

3.1 Context

Web pages may consist of a mixture of media items (text, images, sound, video, flash,...). Most contemporary web browsers try to speed up the browsing process by opening up multiple connections, allowing the browser to make multiple HTTP requests to the web server. Downloading web pages therefore typically generates short, intense bursts of network traffic. Each connection downloads a different part of the web page, which the browser afterwards assembles and presents to the user.

3.2 Network topology

Figure 3 presents the basic topology. All nodes are connected with bi-directional links with a propagation delay of 10ms and a capacity of 10Mb/s for each direction. The overflow strategy for the queues of each link is `DropTail`. The `queue-limit` of the link between node 0 and node 1 is 20.

3.3 Applications

There is one long lasting FTP application that sends data from a FTP server on node 3 to a FTP client on node 2. The FTP application uses the default TCP implementation (i.e. `Agent/TCP`) and is configured to use a sufficient amount of network bandwidth. The window size of the FTP connection (i.e. the `window_` variable of its TCP Agent) is set to 80.

We also simulate the behaviour of another user on the LAN network who is browsing to a website. Every HTTP request results in the transmission of data from a web server to the user. In total, the user makes 3 requests during the simulation. Specifically, there are three short bursts of TCP connections that send data from the web server (node 5) to the user's browser (node 4), while the main FTP connection is transferring data. Every request results in a transfer of 40 files of different sizes from the web server to the user's browser[1] To simulate all requests, we thus require 120 FTP applications that require 120 distinct TCP connections.

- **HTTP requests timings:** The requests start at second 5.0, 10.0, and 15.0 during the simulation. For every request, all 40 files are automatically transmitted within 2 seconds after the request. For example, the first 40 HTTP requests begin transmitting in the time interval [5.0,7.0]. We generated start time intervals for these 40 transfers using a random variable of type `RandomVariable/Exponential` (Poisson distribution), with as average (`avg_`) parameter of the random variable to 0.05. In other words, a new HTTP requests starts approximately

[1]Note we modelled all HTTP requests using an FTP application since support for HTTP is still experimental in the NS2 simulator. Moreover, semantics of the FTP protocol are comparable to HTTP.

0.05s after the previous one. Generating 40 values will give you the time gaps between the start of 2 file transfers (40 × 0.05 is 2, which means that every file transfer will be started within approximately 2 seconds). More explanation of using random variables for inter-arrival times, file sizes, and how to instruct your application to these files can be found in the guide "NS for Beginners", on page 60.

- **HTTP request file sizes:** The size of each file transfer is generated with a Pareto distribution (`RandomVariable/Pareto` of `shape_` 1.5 and `avg_` 150000. For every file, its start time and file size should be written to a file (e.g. `puts $filename $httprequest $starttime $filesize`).

3.4 Logging

Make sure your TCL-script periodically logs the congestion window (`cwnd_`) and slow start threshold (`ssthresh_`) of the TCP connection used by the main FTP (see examples in the manual).

3.5 Questions (45p)

Please include a printed version of your code. This allows us to better understand your simulation results if needed.

1. Investigate the throughput of the main FTP application. Can you indicate the effects the individual bursts of web traffic have on its total throughput? Why do these effects not immediately become visible when each burst starts (i.e. at 5, 10, or resp. 15 seconds)?

2. Plot the congestion window size and slow start thresholds in another graph. Can you identify the different phases of the TCP algorithm? When is the slow start threshold recalculated and how?

3. Discuss the AIMD principle of TCP. Take one 'sawtooth' pattern in the graph and indicate a few reasonable values for the congestion window on the graph. What is the first interval the TCP congestion avoidance algorithm is active?

4. Change the TCP implementation of the main FTP connection into `Reno`. Plot the congestion window and slow start thresholds. Do you spot the differences with the default TCP `Tahoe` implementation? Why does the window sometimes still drop to zero or one?