Project 1: Congestion control

Future Internet

ETH Zürich - Spring Semester 2019

General Future Internet project stipulations:

- This is 1 of 4 projects. Together they account for 50% of your final course grade.
- Always cite and reference appropriately. Do not use other students’ code outside of your group.

This particular project stipulations:

- Submission deadline: **25 March 2019 at 15:00**.
- The project must be done in groups of two.
- You can receive 12.5 points for this project: achieving threshold A (4pt), achieving threshold B (8pt), optimize your solution - up to 12.5 points.
- The group that achieves the best overall performance will be invited to present its solution in-class.
- After the deadline, you will have an interview. Each group member must be able to individually explain and demo all parts.

This assignment has been adopted from Stanford course [CS244](#)
1 Background

The goal of this project is to design a congestion control algorithm that operates well on cellular networks. It is easy to design a congestion control algorithm that is optimized only for packet latency or only for throughput. Being conservative and sending one packet only when an ACK signal is received gives near-perfect latency. On the other hand, sending as much data as possible will maximize the throughput, but in that scenario, the latency will significantly increase. The question is how to balance between these two dimensions.

In this exercise, we optimize the trade-off between throughput and latency in cellular networks where network conditions (i.e. available bandwidth and latency) oscillate frequently. For more information about the problem (and potential solutions) watch the following video (link).

2 Prepare for the contest

First, you should install libraries and tools you need to run the project:

$ sudo apt-get install build-essential git debhelper autotools-dev dh-autoreconf iptables protobuf-compiler libprotobuf-dev pkg-config libssl-dev dnsmasq-base ssl-cert libxcb-present-dev libcairo2-dev libpango1.0-dev iproute2 apache2-dev apache2-bin iptables dnsmasq-base gnuplot iproute2 apache2-api-20120211 libwww-perl

If you do not have a Linux machine or you do not want to do pollute your system, you can us a VM with all dependencies preinstalled (VM link).

To setup the project, do the following:

$ cd project1
$ cd mahimahi
$ ./autogen.sh && ./configure && make
$ sudo make install
$
$ cd ..
$ cd sourdough
$ ./autogen.sh && ./configure && make
$
$ sudo sysctl -w net.ipv4.ip_forward=1

To run the example:

$ cd sourdough/datagrump
$ ./run-contest
The example code runs one client and one server, where the client sends a continuous stream of data to the server. The script varies the bandwidth between the two according to a particular trace. The data is transmitted between the machines using a single link with an unlimited outgoing buffer. It is important to notice that in this scenario, there is no packet loss. For all our experiments, we will use the trace stored in:

```
mahimahi/traces/Verizon-LTE-short
```

We can test your code against a different trace. To change the trace you use, modify the `run-contest` script.

### 3 Visualization

The `run-contest` script provides a visualization of bandwidth and latency over time. Figure 1 (left) shows available bandwidth (red area), send throughput (blue line), and receive throughput (red line). Ideally, two lines should match the edge of the red area. Figure 1 (right) shows queuing delay during the transport. Ideally, it should remain close to zero.

### 4 Submit your solution

After you successfully execute `run-contest`, the trace would be placed under:

```
project1/sourdough/datagrump/current_trace/trace.log
```

After you commit a new version of this file, we will refresh your score on the leader board.

### 5 Contest

Your task is to optimize the trade-off between bandwidth and latency. Your score is calculated as:

```
log(average_throughput/(95_percentile_latency*0.001))
```
You see the score after executing the *run-contest* script. To improve your score, you should focus on changing the congestion control controller placed in:

`project1/sourdough/datagrump/controller.cc`

Every time you change the file, you must rerun the *make* command. In general, if your implementation requires more advanced techniques, you are free to change other files in the *datagrump* folder apart from the *controller*.

### 5.1 Task A

Maximize your score by changing the fixed window size in the *controller*. You should achieve the score higher than **2.48**. Achieving score **2.48** will give you at least 4 points.

### 5.2 Task B

Maximize your score by changing the window size dynamically. Think about additive increase/multiplicative decrease [link].

You should achieve score higher than **3.15**. Achieving the score **3.15** will give you at least 8 points.

### 5.3 Task C

Optimize your solution to reach the top of the leader board.

### 6 Final notice

The maximum number of points is **12.5**. In your algorithm, **do not hardcode the trace**. You can make decision based on the bandwidth you have observed, but not based on the bandwidth changes that will come in the future.

Before the deadline, write a short **readme** (up to 5 sentences) about how your algorithm works and store it in:

`project1/sourdough/datagrump/current_trace/readme.txt`

You can find the leader board [link]