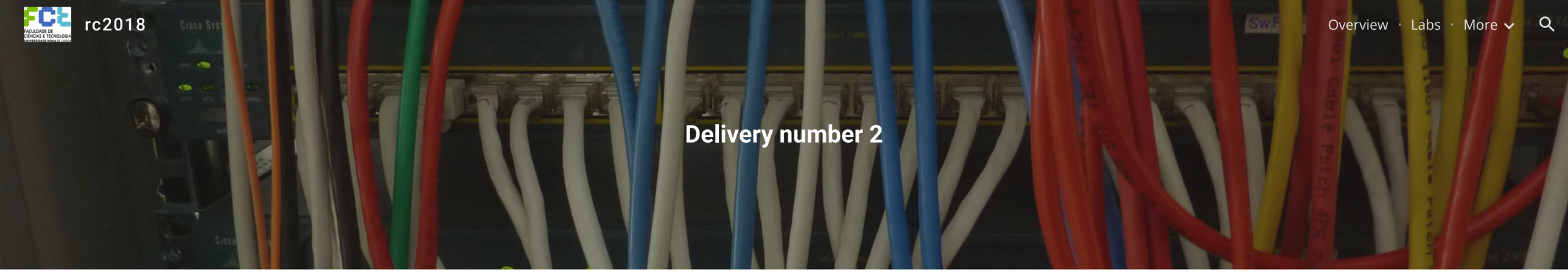

Delivery number 2

Computer Networks

Lab #5 - Delivery number 2

A client to download content from one or more HTTP servers by the way of HTTP range requests

Motivation

In today's internet, most of the users consumed content is carried over the HTTP protocol. In the specific case of multimedia content, the volume of the consumed information varies from a few Mbytes, in the case of photographs, up to several G bytes, in the case of movies.

It is not realistic to think of such bulky objects being transferred in a single HTTP request / reply interaction, using a single TCP connection. Inevitably, due to the high volume of data of such content, momentary anomalies in the network, or problems in the servers, it is necessary to resort to more than one interaction among the client and the server(s). In addition, in the case of films, as they can take several hours to play, it is not mandatory or interesting to transfer in one only chunck the full content, or from the same server.

Most of the high volume content is transfered by different techniques. Transferring by the way of using several HTTP range transfers (see the previous exercise) is a common option.

Goal

To complete this exercise you must program an HTTP client able to transfer a volumonious file (e.g. of size 100 Mbytes) from one or from a set of HTTP "tricky" servers. These servers, accept range HTTP requests, and whenever they receive a request o an object with more than 1 Myte, they only send a slice of the requested object of random size, from 1 Mbyte, up to at most 10 Mbytes. Thus, if the requested object (or range) has less than 1 Mbyte, it is fully sent. However, if the requested object (or range) has size grater than 1 Mbyte, these servers send a range of random size, between 1 and 10 Mbytes, in a random way.

In addition, they also may change randomly the performance of the transfer.

Minimal and optional goals of your work

Program the ***GetFile*** HTTP client able to fully download a (huge) file from a set of HTTP "tricky" servers, which reply with a range of the requested file, of random size, if the requested range (or the full file) is grater than 1 Mbyte. These "tricky" servers send at most 10 Mbytes in each reply. The performance of the connection used in each reply is also random.

Your client should be run in the following way:

```
java GetFile http://server[:port]/file_name
```

For downloading the IFB.mp4 movie trailler your client will run in the following way:

```
java GetFile http://localhost[:8080]/IFB.mp4
```

For your implementation and testing purposes, you can use one only server or a set of four servers (using the HttpTrickyServer.java available in this folder.

```
To start one only server (it runs by default in port 8080):
java HttpTrickyServer

To start a pool of four servers, you must use a different port for each one:
java HttpTrickyServer 8080 &
java HttpTrickyServer 8081 &
java HttpTrickyServer 8082 &
java HttpTrickyServer 8083 &

Note) You can also use the available script to launch the four servers (all in balckground) - See the script
serverclusterstart.sh
```

Then, using the four servers you can serve your client requests in the local host, listening for connections in a different port: 8080, 8081, 8082, 8083.

In your implementation and tests you can implement whatever solution you prefer:

a) Sending a set of successive requests to the same server;

b) Sending requests to several different servers, for example in a round-robin way

c) Sending requests to several different servers, in parallel.

d) Any other policy, according to the evolution of your observations

Note 1: your client only receives one URL as the parameter. Therefore, if your client contacts more than one server, it must assume that the other servers are in the same host, at a contiguous increasing range of ports, as described before.

Note 2: the servers only serve one client after the previous one (they are not concurrent servers, not serving several clients in parallel).

Output statistics

It is mandatory that your program collects the following informations:

- Time elapsed to complete the full transfer (in seconds)
- Total number of bytes downloaded (in bytes)
- End-to-end average bitrate of the full transfer (in bytes/sec)
- Number of requests performed by the client during the file transfer
- **Optional:** average size of the payload of each HTTP reply (in bytes)
- **Optional:** average time spent in each request/reply (in milliseconds)

Use the following output format:

```
Total time elapsed:                .... seconds
Total number of bytes downloaded:  .... bytes
End-to-end average bitrate:        .... bytes per second
Number of requests performed:      .... requests
```

Grading (in 0-20 scale)

- A program that does not transfer the file or transfers it **incorrectly**, will be graded at most 7.
- A program that transfers the file **correctly** to a client local file, only contacting one server at a time, will be graded at most 13.
- A program that transfers the file **correctly**, using the four servers, will be graded at most 15.
- Programs that use more then one server in parallel, have no a priori grading limitations.
- Code clarity and structure, as well as performance indications according to your used policy when using the four servers will be accounted for grading purposes (ranging from 1 to 5).

Programs may be developed in groups of up to two students. However, they must be delivered **individually** and will be graded **individually**.

Delivery of your work: delivery form/questionaire here.

Rules:

- The client can be completed and tested by a student or by a group of up to two students. However, each student **must** deliver his own copy of his/her (or the group) results and files.
- All students must present a demo of the implemented client in the corresponding laboratory class. The demo will be registered in a demo form by the instructor). This will take place from THU (8/Nov/18) to WED (14/NOV).
- On THU (14/NOV), 17h00, a final form (Google From) will be available for the final submission of your work. In this form you ill find a final testing environment for your implementation and you must register the observed results (similar to this filled form example in pdf) for that testing conditions. The allows the identification of the other member of the group if he/she exists, as well the submission of your final client code.
- **Important:** The final submission, using the previous form is only possible on THU (15/Nov), from 0h00 to 24h00.

Additional materials

All the needed materials (testing files, programs, scripts, ...) are available in this folder.