Socket exercise

Daniel Hagimont

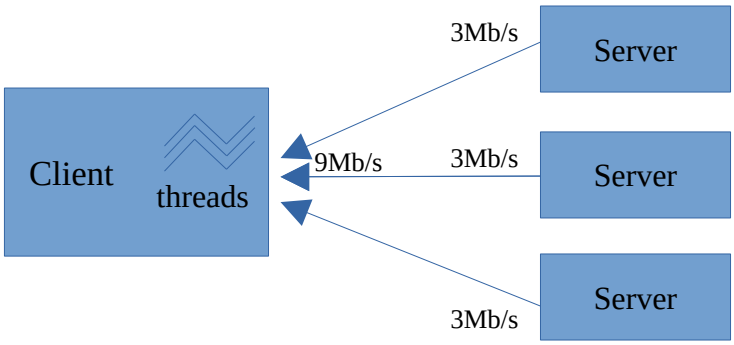
Daniel.Hagimont@irit.fr

USTH

March 2024

The objective of this exercise is to program with sockets.

We are considering an asymetric network such as ADSL (Asymmétric Digital Subscriber Line) in France. In such networks, the bandwidth in upload is much lower than the bandwidth in download. This implies that when a Client downloads a large document, provider by a Server, the data transfer is limited by the upload bandwidth from the Server. In order to reduce this transfer time, it is possible to run in parallel the download of different fragments of the large document from different servers which store a copy of the document. This technique is used by many peer-to-peer platforms.
For instance, if the upload bandwidth is 3Mb/s and the download bandwidth is 10Mb/s, the Client downloading with one Server is limited to 3Mb/s while downloading in parallel with 3 Servers allow downloading at 9Mb/s. Obviously the client has to be multithreaded to download in parallel from different Servers.



We want to implement a similar communication scheme with sockets.
You have to implement 2 socket based programs which are executed respectively on the Client and Server machines :

- Client : the program which downloads in parallel all the fragments from several Servers
- Server : the program which executes on each Server and which provides fragments.

We suppose that both programs Client and Server have the following variables defined :

```
final static String hosts[] = {"host1", "host2", "host3"};
final static int ports[] = {8081,8082,8083};
final static int nb = 3;
static String document[] = new String[nb];
```

The 3 first variables give the addresses (@IP, #port) of Servers.
A Server starts with its server number (0, 1 or 2 which is its index in tables hosts and ports).
We are managing a single document which is represented here with a table of Strings, each String is a fragment. We assume that this table is initialized for the Server program (so each Server has a copy of the document). This table has to be filled by the download for the Client.
For requests, you can used object serialization. The client can simply send on a TCP connection the number of the fragment he wishes to download and receive on that connection the String which represents the fragment.

Implement the 2 programs Client and Server following this scheme.