

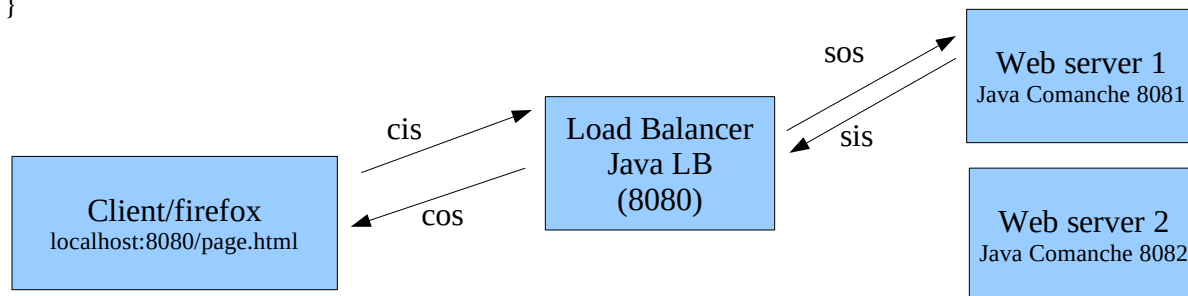
# Socket lab

Daniel Hagimont (daniel.hagimont@irit.fr)

The objective of this lab is to practice with Socket programming in Java. You have to implement a load balancer which receives HTTP requests and forwards them towards a set of web server instances, following a random policy.

The beginning of the class you have to implement is:

```
public class LoadBalancer {  
    static String hosts[] = {"localhost", "localhost"};  
    static int ports[] = {8081,8082};  
    static int nbHosts = 2;  
    static Random rand = new Random();  
    ...  
}
```



Whenever an HTTP request is received (a new TCP connection), *LoadBalancer* forwards the request to one of the web servers (the servers' addresses are given by the *hosts* and *ports* tables), and it forwards the result from the web server back to the client. The choice of the web server is random (*rand.nextInt(nbHosts)* returns an integer between 0 and *nbHosts-1*). For efficiency, *LoadBalancer* is multithreaded.

For input/output programming, we will use:

- InputStream
  - public int read(byte[] b); // blocking, returns the number of read bytes
- OutputStream
  - public void write(byte[] b, int off, int len); // write "len" bytes from offset "off"

In this lab, we suppose that requests and responses can be read and written with a single method call (read() or write()) with a buffer of 1024 bytes. Notice that this assumption is not valid in the real life.

To test your LoadBalancer, you can use a simple web server (Comanche.java).

You can run 2 instances of Comanche from 2 terminals:

```
java Comanche 8081  
java Comanche 8082
```

I assume that your LoadBalancer accepts connection on port 8080. You can launch your LoadBalancer from a 3rd terminal.

Then, with a web browser, use the URL: localhost:8080/page.html  
(page.html is supposed to be present in the directory from where you ran Comanche)