System Programming
8th Laboratory (26th and 28st April 2017)

The architecture of the project (Client/Gateway/Peers is similar to that of the last laboratory).
Adapt the code done on the last laboratory to start developing the project.

Take attention to the following:

- students should start implementing the library (following the API specification)
- threads can now be used
- on the gateway two different sockets can be used (one to receive requests from the client and another to receive information from the peers) and two threads should be used to handle each of these sockets
- The peers should be mutithreaded (so that multiple clients can connect simultaneous). After the accept a thread should be created.

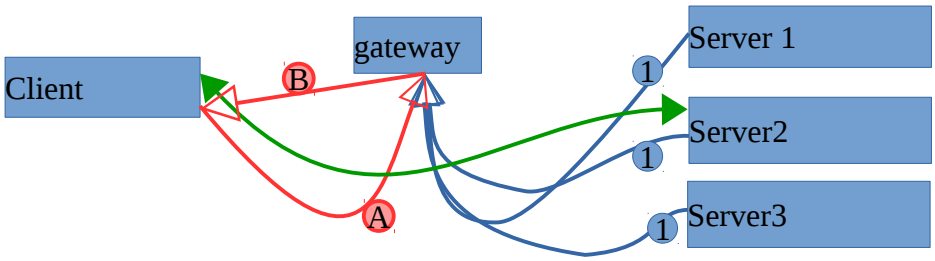
Gateway/load balancer implementation

This laboratory will exercise the use of Sockets and multiple processes to implement a load balanced system.

Student should implement 3 different type of processes:

- Gateway
- Server (simple echo server)
- Client (adaptation of the client from previous lab)

When a client wants to interact with a server, first contacts the gateway that will forward it to one of the available servers.
The client will then interact with the server until it decides to disconnect.
Each server sends messages to the gateway announcing its address (that will be sent to the client).
The architecture of the system is as follow:



Sockets in blue and read are datagram, sockets in green are stream.

Server

Each server has a socket (binded to a certain address) that receives connections from the clients. The address of that socket is sent to the gateway when the server starts (message (1)) .

The server socket receives messages from the clients. The messages are text strings that the server prints on the screen. As response, the server sends back the message in all UPPER CASE.

After a client connects a thread should be created to handle all the client communication. The threads should be terminated when the client disconnects.

Gateway

The gateway only has two datagram sockets

- one receives messages (1) from the servers
- another receives messages (A) from the clients.

Thow therads should be created (one responsible for reading on each socket).
The gateway maintains a linked list with the identifications of all available servers. The list is updated when a server sends a message (1):

- When a server starts running, it contacts the Gateway to inform its address.

When the gateway receives a message (A) from a client, it searches on the linked list for the next server and sends as response the address of that server (message (B)). The selection of the server follows a **round robin** policy.

Client

The client will read the address (string) and the port of the gateway from the keyboard, contact the gateway sending one message (A) through a datagram socket and will read the address of the server.
This address will be used to connect a stream socket. ~~Afterward starts interacting with it:~~

- ~~read a string from the keyboard~~
- ~~send the string to the server~~
- ~~read a response~~
- ~~print response on the scren~~

Students should start implement the API defined in the project assignment.

Message formats

Define the message structures needed in the project.
~~Messages exchanged between the client and the servers are strings that have a maximum length of 100 bytes.~~
~~Since the gateway receives messages from two different clients it is necessary to distinguish between them. The simplest way is to define the following structure:~~

```
typedef structure message_gw{
    int type;
    char address[20];
    int port;
}
```

~~The only difference between the messages sent by the servers and the clients is the type field: for instance 0 for message (A) and 1 for message (1).~~
~~This same structure can be used as message B. If there is no server available, the type value can be 0, otherwise 1.~~

API information

- http://www.gnu.org/software/libc/manual/html_node/Sockets.html
- http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html
- http://pubs.opengroup.org/onlinepubs/9699919799/functions/V2_chap02.html#tag_15_10
- http://beej.us/guide/bgnet/
- http://tldp.org/LDP/lpg/node7.html

SOCKETS

- man socket
- man 2 bind
- man 2 accept
- man 2 connect
- man unix
- man 7 signal