# **Implementing Lamport Clock in GO**

## **Problem Statement**

The statement below is just a random thought of the top of my head, just to give some context to the usage of Lamport Clocks. Consider this:

“An e-educational website in which a professor teaches a topic which is divided into subtopics. There are students listening to this lecture live. If a student has a question it posts it to the server. Later when the lecture is over, the professor takes a look at the questions in some file (let’s say log) and decides the topics to which each question belongs (Let’s assume he’s not a good professor and can’t decide just by looking at the questions :P). We can implement a Lamport Clock to make sure that the students question falls into the right category in the logs”.

## **Implementation**

There are 3 files

* Server.go – the hosting application.
* ServerHelper.go – is assumed to be some code that decides when the next topic has started.
* Client.go – is the user.

Now, the client contacts host with the question. As the client time may be different from the server time, a Lamport Clock is implemented to make sure that the questions synchronize with the topic going on. Just for fun, the serverhelper has its own Lamport clock and clock at the Server is managed by the helper’s and the client’s clocks.

Every time the clocks are in sync, an “Increment” happens, meaning that the clock at the server has incremented. In cases there are synchronization issues, the clock at server gets “Update”.

## **User Guide**

Same Machine :

* Start the server
* Start the client
* Start the serverhelper – this is gonna kill the server after 20 seconds, and gonna keep incrementing server’s clock by 1.
* With all three running, use client to keep asking questions. After 20 seconds type “exit” at questions. After server shuts down, typing anything else will give connection error.
* After server shuts down, check the “Logs.txt” file in the same directory with the server file. Each question belongs to the topic before it.

Different Machine:

* Start server with Server IP as an argument
* Start client on a different machine with client IP as an argument. Make the (client machine time > server machine time), and later compare client command line with logs
* Start server helper with Server IP as an argument

You can ask questions from client even when severhelper is not running, in case some extensive testing is needed (because with serverhelper active, everything shuts down in 20 seconds).

Further to see effective lamport implementation, change the global clientLamportClock variable at client.go file, just to see the effects produced.

References :

|  |  |
| --- | --- |
| <https://gist.github.com/ijt/950790> | Using http.GET method in golang |
| <https://golang.org/pkg/net/url/#example_URL> | Using query strings in GO to communicate |
| <https://blog.golang.org/go-maps-in-action> | Using GO maps |
| https://golang.org/doc/articles/wiki/ | General help for making a web application in GO |