Report 3: Loggy

Gabriele Morello

September 28, 2022

1 Introduction

In this homework I wrote a logger to register messages sent from different nodes, we also want to order events before printing them, to do that I used two strategies: Lamport clocks and vector clocks. Some of the modules were given so I will focus on the modules loggy and time also called vect in the vector implementation. The other models, worker and test, are reported as given in the instructions.

2 Main problems and solutions

I will now go through the time and the vect module to explain how I implemented them and show differences, later I will explain the loggy module.

2.1 Lamport clock

I implemented the time as an integer, while the clock is a list of tuple where each element contain a node and an integer.
Some functions in the time module are trivial,

- zero() returns 0,
- inc(Name, Time) increments Time by 1,
- merge(Ti, Tj) takes the maximum value between two timestamps
- leq(Ti, Tj) returns a boolean based on Ti less than equal Tj.
- clock(Nodes) creates the clock using a foldl call, the return value is the list of tuple described above,
- update(From, Time, Clock) replaces the entry having From as a key, in the Clock list, with a new tuple with elements From and Time, before returning it sorts the clock list to ensure that the oldest timestamp always comes first. The last function is

- safe(Time, Clock) whose task is to check that the Time argument happened before the first event in the Clock, it does so by calling leq with Time and the first element of Clock as arguments.

2.2 Vector clock

With Lamport we couldn't have causal order, to achieve that we can implement a different kind of clock represented as a vector. In Erlang we do that using a list of tuples where each tuple contains a Node and a list of tuple with a Node and the respective timestamp. Time is a list of tuples with Node and timestamp in each entry.

- zero() returns an empty list,
- inc(Name, Time) search for the given Node in Time list and if found increment its value by 1, if not found add the Node to the list with initial value of 1,
- merge(Ti, Tj) merges to timestamps iterating through the first one and comparing it to matching values in the second one if there are collisions we take the maximum value otherwise we add the new element,
- leq(Ti, Tj) search for the first element in Ti in Tj list and, if it founds it, it compares them in case the first one is less than or equal the second it calls itself recursively otherwise it returns false.
- clock(Nodes) returns an empty list,
- update(From, Time, Clock) replaces the entry having From as a key, in the Clock list, with a new tuple with elements From and Time, if the From node is not presented in Clock we add it and Time will be its value.
- safe(Time, Clock) iterates through Clock and for each element it calls leq to compare Time with an entry of Clock.

2.3 logger

In the logger module I had to implment the loop function in this function when I receive a message I update the clock, I sort the queue by time with the new message and call a recursive function rec_check that goes through the queue and checks if it can log any elements, if it can it calls the log function while if it can't it returns the queue. loop is recursive and can be stopped with a stop message.

3 Evaluation

Using Lamport clocks I achieved the same end result as when I used vector clocks but vector clocks should be more flexible. Unfortunately I'm not been able to test this flexibility, because I whenever I tried to start a new node while workers where running I always got errors. However I successfully tested the scenario where a node dies and the system kept running, as a result I had a very long queue at the end.
I paste below results from the Lamport run and the vector run.

```
log: 20 ringo {sending,{hello,896}} [{ringo,20},{john,20},{paul,21},{george,22}]
log: 20 george {received,{hello,381}} [{ringo,20},{john,20},{paul,21},{george,22}]
log: 20 john {received,{hello,943}} [{ringo,20},{john,20},{paul,21},{george,22}]
log: 21 paul {received,{hello,896}} [{paul,22},{george,25},{john,26},{ringo,27}]
log: 21 john {sending,{hello,290}} [{paul,22},{george,25},{john,26},{ringo,27}]
log: 22 george {received,{hello,290}} [{paul,22},{george,25},{john,26},{ringo,27}]
log: 22 john {sending,{hello,952}} [{paul,22},{george,25},{john,26},{ringo,27}]
log: 22 paul {sending,{hello,392}} [{paul,22},{george,25},{john,26},{ringo,27}]
log: 23 paul {sending,{hello,825}} [{paul,23},{george,26},{john,27},{ringo,27}]
log: 23 george {received,{hello,952}} [{paul,23},{george,26},{john,27},{ringo,27}]
log: 23 john {received,{hello,392}} [{paul,23},{george,26},{john,27},{ringo,27}]
log: 24 george {sending,{hello,319}} [{george,27},{ringo,27},{john,28},{paul,29}]
log: 25 john {received,{hello,319}} [{george,27},{ringo,27},{john,28},{paul,29}]
log: 25 george {sending,{hello,521}} [{george,27},{ringo,27},{john,28},{paul,29}]
log: 26 john {received,{hello,521}} [{george,27},{ringo,27},{john,28},{paul,29}]
log: 26 george {sending,{hello,649}} [{george,27},{ringo,27},{john,28},{paul,29}]
log: 27 john {received,{hello,825}} [{george,27},{ringo,27},{john,28},{paul,29}]
log: 27 ringo {received,{hello,649}} [{george,27},{ringo,27},{john,28},{paul,29}]
log: 27 george {sending,{hello,885}} [{george,27},{ringo,27},{john,28},{paul,29}]
log: 28 george {sending,{hello,202}} [{george,28},{ringo,28},{john,29},{paul,29}]
log: 28 ringo {sending,{hello,265}} [{george,28},{ringo,28},{john,29},{paul,29}]
log: 28 john {received,{hello,885}} [{george,28},{ringo,28},{john,29},{paul,29}]
log: 29 john {received,{hello,202}} [{paul,29},{john,30},{george,31},{ringo,32}]
log: 29 paul {received,{hello,265}} [{paul,29},{john,30},{george,31},{ringo,32}]
```

```
[{george,[{george,80},{paul,84},{john,90},{ringo,77}]},{paul,[{paul,85},{john,92},{ringo,75},{george,78}]},{john,[{john,92},{ringo,75},{paul,84},{george,78}]},{ringo,
[{ringo,76},{john,90},{paul,84},{george,79}]}]
log: [{paul,82},{john,87},{ringo,72},{george,76}] paul {sending,{hello,598}}
[{george,[{george,80},{paul,84},{john,90},{ringo,77}]},{paul,[{paul,85},{john,92},{ringo,75},{george,78}]},{john,[{john,92},{ringo,75},{paul,84},{george,78}]},{ringo,
[{ringo,76},{john,90},{paul,84},{george,79}]}]
log: [{paul,83},{john,90},{ringo,73},{george,78}] paul {received,{hello,601}}
[{george,[{george,80},{paul,84},{john,90},{ringo,77}]},{paul,[{paul,85},{john,92},{ringo,75},{george,78}]},{john,[{john,92},{ringo,75},{paul,84},{george,78}]},{ringo,
[{ringo,76},{john,90},{paul,84},{george,79}]}]
log: [{paul,84},{john,90},{ringo,73},{george,78}] paul {sending,{hello,90}}
[{george,[{george,80},{paul,84},{john,90},{ringo,77}]},{paul,[{paul,85},{john,92},{ringo,75},{george,78}]},{john,[{john,92},{ringo,75},{paul,84},{george,78}]},{ringo,
[{ringo,76},{john,90},{paul,84},{george,79}]}]
log: [{ringo,74},{john,90},{paul,84},{george,78}] ringo {received,{hello,90}}
[{george,[{george,80},{paul,84},{john,90},{ringo,77}]},{paul,[{paul,85},{john,92},{ringo,75},{george,78}]},{john,[{john,92},{ringo,75},{paul,84},{george,78}]},{ringo,
[{ringo,76},{john,90},{paul,84},{george,79}]}]
log: [{ringo,75},{john,90},{paul,84},{george,78}] ringo {sending,{hello,421}}
[{george,[{george,80},{paul,84},{john,90},{ringo,77}]},{paul,[{paul,85},{john,92},{ringo,75},{george,78}]},{john,[{john,92},{ringo,75},{paul,84},{george,78}]},{ringo,
[{ringo,76},{john,90},{paul,84},{george,79}]}]
log: [{john,91},{ringo,75},{paul,84},{george,78}] john {received,{hello,421}}
[{george,[{george,83},{paul,87},{john,93},{ringo,81}]},{paul,[{paul,86},{john,92},{ringo,77},{george,81}]},{john,[{john,93},{ringo,75},{paul,84},{george,78}]},{ringo,
[{ringo,80},{john,93},{paul,87},{george,82}]}]
log: [{john,92},{ringo,75},{paul,84},{george,78}] john {sending,{hello,537}}
[{george,[{george,83},{paul,87},{john,93},{ringo,81}]},{paul,[{paul,86},{john,92},{ringo,77},{george,81}]},{john,[{john,93},{ringo,75},{paul,84},{george,78}]},{ringo,
[{ringo,80},{john,93},{paul,87},{george,82}]}]
log: [{john,93},{ringo,75},{paul,84},{george,78}] john {sending,{hello,965}}
[{george,[{george,83},{paul,87},{john,93},{ringo,81}]},{paul,[{paul,88},{john,94},{ringo,77},{george,81}]},{john,[{john,93},{ringo,75},{paul,84},{george,78}]},{ringo,
[{ringo,80},{john,93},{paul,87},{george,82}]}]
```