Report 3: Loggy - A logical time logger

Bernardo González Riede

September 28, 2017

1 Introduction

As mentioned in the assignment paper, the intended outcome is lo learn through a practical example how Lamport clocks and vector clocks work.

2 Main problems and solutions

2.1 Questions in the report

Chapter 3: How do you know that [the messages] are printed in the wron order?
Looking at the Lamport clock timestamp and the 'random' value we can see if a received message was printed before the correspoinding sendig message. What is always true and what is sometimes true? How do you play it safe?
To play it safe, the logger hast to wait until it receives a timestamp from every node which is higher the the message to output.

3 The time module

The time module for the Lamport clock implementation is actually a very simple one.

- Zero/0 returns 0 for initializing the time value for a given node.
- Inc/2 sums 1 to the time given.
- Merge/2 returns the higher value of the provided times using the erlang:max/2 function.
- Leq/2 uses a comparision to return true or false.
- Clock/1 returns a list of tuples with length N and composed of the names provided with a 0 to start with.
- Update/3 retrieves the existing entry first to be able to compare the existing time and updates it only if the provided time is higher.

- Safe/2 was implemented with 3 cases. Essentially it tries to loop over the clock provided. It stops if the time provided is higher then an entry in the clock and returns false. Otherwise it will eventually reach a point where the clock is empty. At that point safe/2 returns true.
- NewTime/2 was introduced to make the other modules less aware of time itself. When receiving a message, the node only cares about the new time, which is generates through merge/2 and a consecutive leq/2, so this function calls both at once.

4 The vector module

The vector module makes things more interesting adding certain complexity.

- Zero/0 returns an empty list.
- Inc/2 gets a bit more complicated too, since it has to search for its own entry in the vector clock. It searches first if there's an entry and sums 1 to it, if existent; if not, it'll create an entry.
- Merge/2 tries to find an existing entry and compares it with the provided value to only store the higher one. If no entry exists, the provided entry is stored.
- Leq/2 compares all the time entries from a given time stamp with a given clock. It iterates over them until one of the provided values is higher or all the time entries have been compared and found to be less or equal to the values in the clock. In the latter case it returns true, since the provided time stamp is older than the current clock. In the former case the timestamp is newer than the clock.
- Update/2 searches first for entry corresponding to the node in the timestamp given and then searches for an entry from the same node in the clock to be able to update it. If no entry is found in the clock, the entry retrieved from the time stamp is stored.
- Safe/2 iterates over the given time stamp comparing the values of the entry with the ones from the clock. It continues as long as the time stamp value is less or equal than the value in the clock. If corresponding value in the clock is found, or the comparision failes, safe/2 returns false.