Report 3: Loggy

Alan Goran

September 27, 2017

1 Introduction

This assignment was about keeping track of logical time and use it in practice to illustrate its effect and importance. The task itself was to implement a logging procedure that receives messages from a number of workers, where the messages were tagged with a time stamp to keep the messaging between the workers in order.

2 The task

The task here may sound very straight forward but there are much more to it than one would think. There were three main parts to implement to successfully do the task, the logger, the worker, and the Lamport time.

The logger is a module that accepts messages and prints them out in the right order. It is doing that by adding messages to a queue and pop them out safely (in FIFO order). By safely it is meant to make sure that the messages that arrives at a node is not an old message. The logger acts like an intermediate node in between the worker nodes.

The worker module is feeding the logger with information. It sends the messages to the logger along with a Lamport time stamp to keep track of what is happening and that things are dine in order. It also adds a Jitter, a slight delay between sending the message to the peer and informing the logger. The purpose of the Jitter is to make the process be more like the reality, the reality being different virtual machines. If we do not use the Jitter we will not have messages occurring out of order when we are running on the same virtual machine.

Lastly, the Lamport time module is implemented in order to add logical time to the worker process. It is keeping track of its own counter and passing this along with any message that is sent to the workers.

3 Main problems and solutions

The main problem, in the assignment, was the logger module and to get the messages sent and received to the workers in the right order, since the messages are received by the logger in a mixed order. Getting this right was a fairly hard challenge. The solution to it was to save the messages in a buffer (hold-back queue) and sort it according to the time stamp. After that we could easily pop the messages in the right order (if the stamping of the messages we done correctly). The logger module had also an internal clock in order to decide which is the message to be printed out. If a message's time stamp was smaller than the internal clock then the message was okay to print out.

4 Evaluation

Two implementations were done in this assignment to evaluate. The first one was logging without using the Lamport time and for the second implementation the Lamport time and logic time for sorting is used.

4.1 First Implementation

In this test the messages were completely out of order when I ran the program with four workers and the Sleep and Jitter values set to 500 and 100, respectively. The wrong order of messages can be seen in the results below where it is clear that the received messages are logged before the sending of the message and their orders are all mixed up.

```
log: na george {received,{hello,18}}
log: na ringo {received,{hello,79}}
log: na paul {sending,{hello,79}}
log: na john {sending,{hello,18}}
log: na ringo {received,{hello,33}}
log: na john {received,{hello,95}}
log: na paul {sending,{hello,33}}
```

4.2 Second Implementation

When the Lamport stamping time and logical time logging is used the result becomes correct and in the right order, see below. Here, we make sure that all the messages are flushed out in their stamped order.

```
log: 0 john {sending,{hello,57}}
log: 0 paul {sending,{hello,68}}
log: 0 george {sending,{hello,58}}
```

```
log: 1 ringo {received,{hello,57}}
log: 1 ringo {sending,{hello,77}}
log: 1 george {sending,{hello,100}}
log: 2 john {received,{hello,77}}
log: 2 john {sending,{hello,90}}
log: 3 paul {received,{hello,90}}
log: 3 ringo {received,{hello,68}}
log: 3 paul {sending,{hello,40}}
log: 4 ringo {received,{hello,58}}
log: 4 john {received,{hello,40}}
log: 4 ringo {sending,{hello,42}}
log: 5 john {received,{hello,42}}
log: 5 john {sending,{hello,84}}
log: 6 paul {received,{hello,84}}
log: 6 ringo {received,{hello,100}}
```

5 Conclusions

From this assignment, we can conclude that the Lamport stamping time is working very well for synchronizing multiple worker nodes with each other. This task gives you a rare insight on such a problem and it was eye opening for someone like myself who never really were aware of the existence of this problem.