Loggy: a logical time logger

Nikhil Dattatreya Nadig

September 28, 2017

1 Introduction

The objective of this exercise was learn how to use logical time in a practical example.

The task is to implement a logging procedure that receives log events from a set of workers. The events are tagged with the Lamport time stamp of the worker and the events must be ordered before written to stdout.
The objective of this assignment is to:

- Understand the need for Logical Time.
- Understand Lamport Timestamp Logic.

2 Main problems and solutions

There are three modules for this problem - logger, worker and time. The logger accepts events and prints them on the screen. The worker waits for a while and then sends a message to one of its peers. While waiting, it is prepared to receive messages from peers so if several workers run and connect them with each other it will have messages randomly being passed between the workers. The task is to implement the logical time so that the messages are in order.

```
log: john na {received,{hello,43}
log: paul na {sending,{hello,43}}
```

This denotes the message from paul to john had been received before paul sent this. This is obviously wrong and rectify this we need to add logical timing and print messages only after the messages has actually been received.

3 Evaluation

Since the messages are not being printed in the right order, time stamps are introduced to every message that is being sent. In the *logger* module,

all the messages are logged in and stored. In the *worker* module, messages are sent from one node to another in a random order. In the *time* module, functions for storing the incoming messages, sorting them and printing them only when it is safe to print them are present.

While only the messages that are safe to print are logged, there are several messages that are already in the queue to be printed. In my testing, while having minimum jitters and sleep duration the maximum number of messages in queue to be printed were 16.

```
log : paul 1 {sending,{hello,49}}
log : john 1 {sending,{hello,54}}
log : ringo 2 {sending,{hello,13}}
log : george 2 {received,{hello,49}}
log : ringo 2 {received,{hello,54}}
log : george 2 {sending,{hello,58}}
log : john 3 {sending,{hello,80}}
log : paul 3 {received,{hello,58}}
log : john 3 {received,{hello,13}}
log : paul 3 {sending,{hello,8}}
log : ringo 4 {sending,{hello,59}}
log : john 4 {received,{hello,8}}
```

The output after implementing logical time and printing the messages once the messages have been received. This results in ordered outputs. The output of a certain message being received is always after the message has been sent.

4 Conclusions

We have understood the importance of logical time in systems. Since, achieving same perfect physical time across different machines is almost impossible, this form logical time is as close to accuracy we can get.