Report 3: Loggy

Peter Hamberg

September 22, 2011

1 Introduction

Implement a logging system using lamport timestamps.

This lab is targeted toward the understanding of time and synchronization handling in a distributed system. A logger process receives incoming messages from several worker processes. The messages are timestamped using a logical clock.

2 Main problems and solutions

The key part of the lamport timestamp system can be found in the logger. The logger will hold onto the incoming messages until they are deemed 'safe' to publish.

A message will be considered 'safe' when its timestamp is lower than the maximum timestamp reported from every worker. In order to keep track of this, every incoming message is sorted into a list based on which worker sent it. Since every new message is added to the head of the list and every worker only ever increments its logical clock (monotonicity, yay!) the head of the list will always be the largest value known for that worker's clock. Calculating the correct value for the 'safe' timestamp becomes a simple matter of calculating the minimum value of all the heads.

When the 'safe' timestamp has been decided (this may be 0 if one or more lists are empty), all messages are traversed and any matching the sending criteria are pruned from the lists and logged to stdout.

I encountered some syntactical problems while coding for this assignment, mainly because I wanted to use anonymous functions with clauses, like so

```
From = worker,
lists:map( (fun ({From, List}) -> {From, [{Time,Msg}|List]};
({Name, List}) -> {Name, List}
end).
```

which of course wont work, since where I wanted pattern matching, Erlang instead shadowed the value of the variable. ('From' in this case.)

Another problem that haunted me for a long time was my mistaken notion that I could declare safe any message with a timestamp smaller than - or equal to - the safe timestamp. After correcting this bug the out-of-order log posts finally disappeared.

3 Evaluation

Before implementing the lamport timestamps, the log would often show the receival of any arbitrary message before it logged its sending. With the timestamps implemented, the proper order of the output was achieved.

4 Conclusions

I appreciated this lab, since it forced an understanding of this weeks topic. The actual inner workings of the logical clocks can be difficult to grasp at first, so it is good to be able to implement it in a practical setting. This gives a new angle of observation and helped me grasp some of the less understandable, but very useful aspects of the topic.