Loggy A Logical Time Logger

Jinjia

September 16, 2023

1 Introduction

This assignment is to implement a logical time logger that prints the correct order of events sent by different worker nodes, especially in the distributed system without a globally synchronized clock. The event was tagged with a Lamport time stamp to guarantee partial order. In the end, I used the vector clocks to guarantee total order.

2 Main problems and solutions

When I first tried to do the Lamport time, I didn't know how to make messages print safely, it means messages must be printed in order. But the assignment gave me some hints to do it using two important data structures, a clock keeping track of the timestamps of the last messages seen from each worker and a hold queue keeping log messages that are still unsafe to print. In Lamport time implementation, **clock** procedure is straightforward to initialize the clock data structure. As for **update** procedure, the most important thing is to sort the clock based on the counter after updating to make **safe** procedure easy. If time is less than the minimum time seen by all workers, then it is safe. However, one disadvantage of this method is that if one worker's counter is always zero, the logger doesn't print anything, and the hold queue will become larger and larger.

When it comes to Vector time, every worker has a clock overview of other workers. if we know timestamps, we can easily compare them. If each of its entries is less than or equal to the entries of the other timestamp, this message happens before other messages.

3 Evaluation

The first implementation was without logical time. So the logger would print messages in the wrong order. When I decrease the jitter, the number of wrong entries becomes fewer. Figure 1 is the result when I executed

test:run(2000, 1000). The obvious problem is that one message should be received after it is sent.

The second implementation was with Lamport time. Figure 2 is the result. One problem is that the hold queue is kind of large.

The third implementation was with Vector time. Figure 3 is the result. The hold queue is smaller, compared to Lamport time.

4 Conclusions

The implementation of Lamport clocks and Vector clocks in Erlang has been a valuable learning experience. First of all, I have a deep understanding of handling concurrency and message passing. Second, Having some basic understanding of Lamport clocks and Vector clocks in a distributed system without a globally synchronized clock is beneficial to understand other popular open source projects, such as etcd.

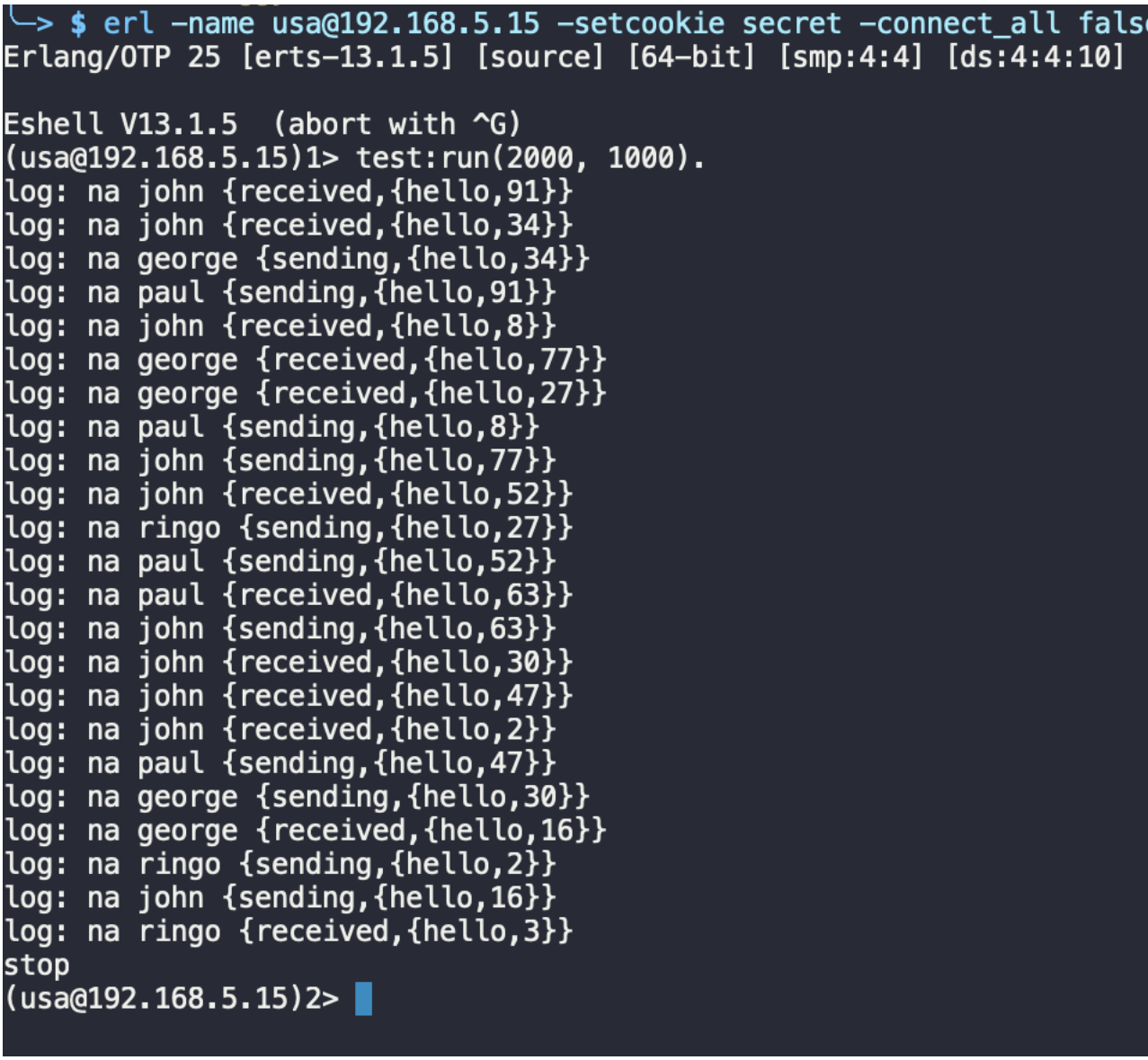

Figure 1: Wrong order

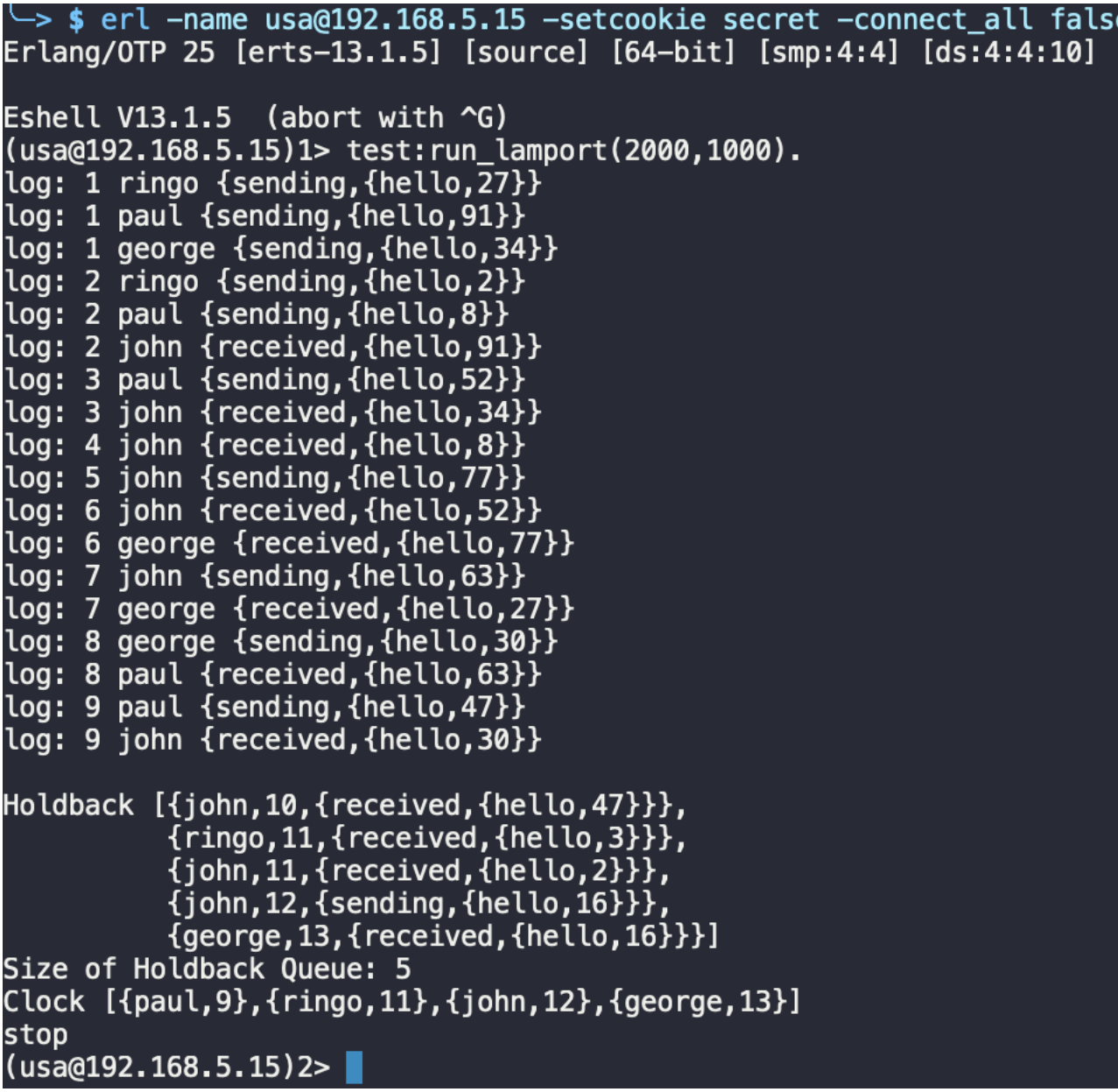

Figure 2: Lamport time

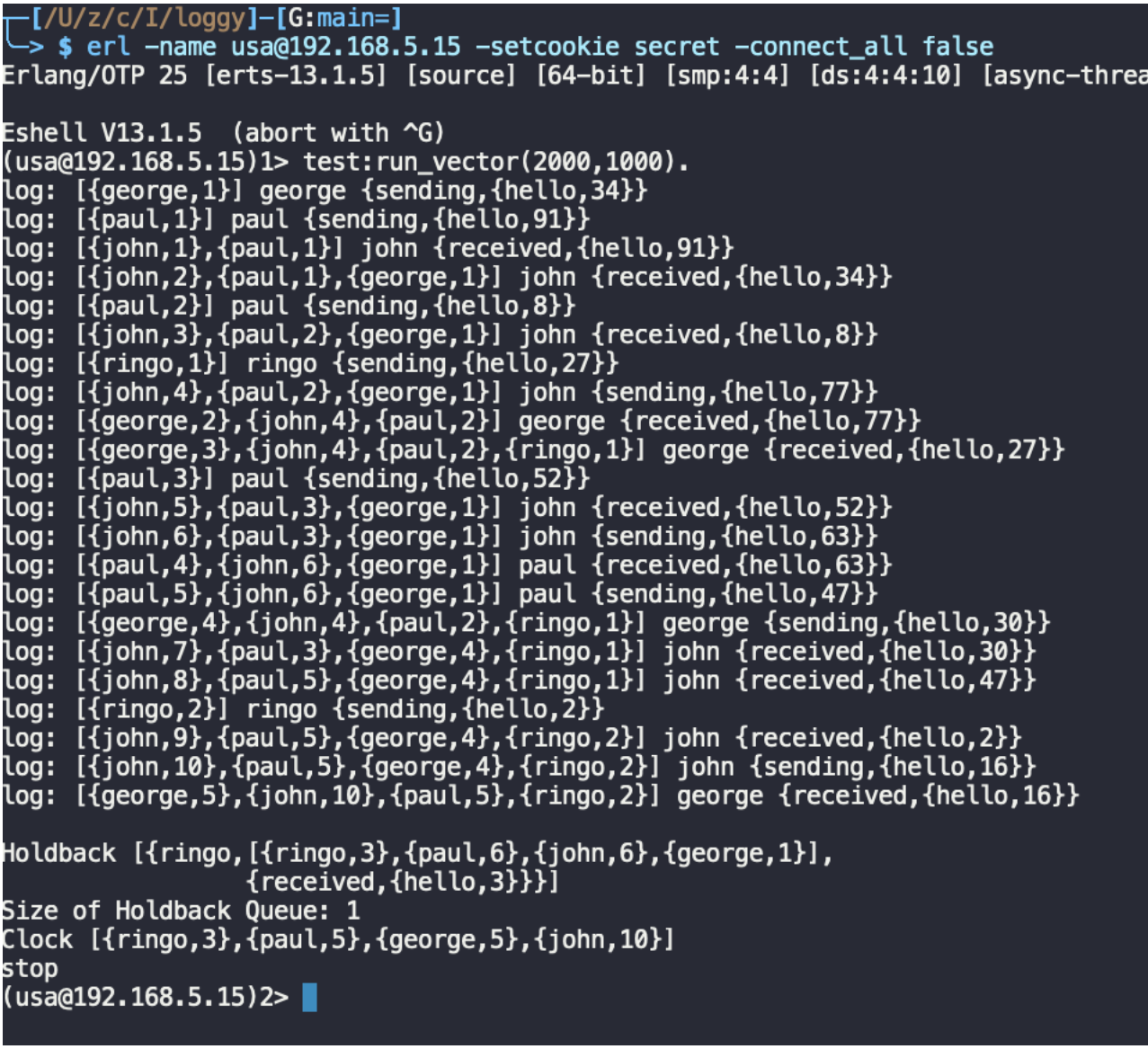

Figure 3: Vector time