Report 3: Loggy

Chen Sihan

September 24, 2020

1 Introduction

The goal of this project is to implement a logical clock by using Lamport and Vector Clock algorithm.

2 Main problems and solutions

Firstly I implement the Lamport version of the logical clock. In the experiment 4 processes are built to send and receive messages from each other. The program should make sure that $A- > B => Clock(A)- > Clock(B)$ is promised (However, the inverse is not promised). In other words, when A send message to B, it must be promised that the logger should record that the information that A sending message comes earlier than B receiving.

To accomplish it, a holdback queue is built to keep all the concurrent processes. Only when the received message clock is larger than any of the clock of the processes in the holdback queue and the clock is not larger than any of the latest clock of processes, can the log information be printed (intuitive way to understand the step).

2.1 Result

As a result, as can be seen in the graph, the send message is printed before receiving message, no matter what value you take for Jitter or Sleep, the order will always be correct.

3 Bonus: Vector

I write 4 corresponding modules for the Vector version of the logical time. The data structure I use here is different from the given pdf. I just use $[0, 0, 0, 0]$ vector for the clock representation for four processes. For example, when the first process increment by 1, the vector should be $[1, 0, 0, 0]$. In a similar way, the output should be like following:

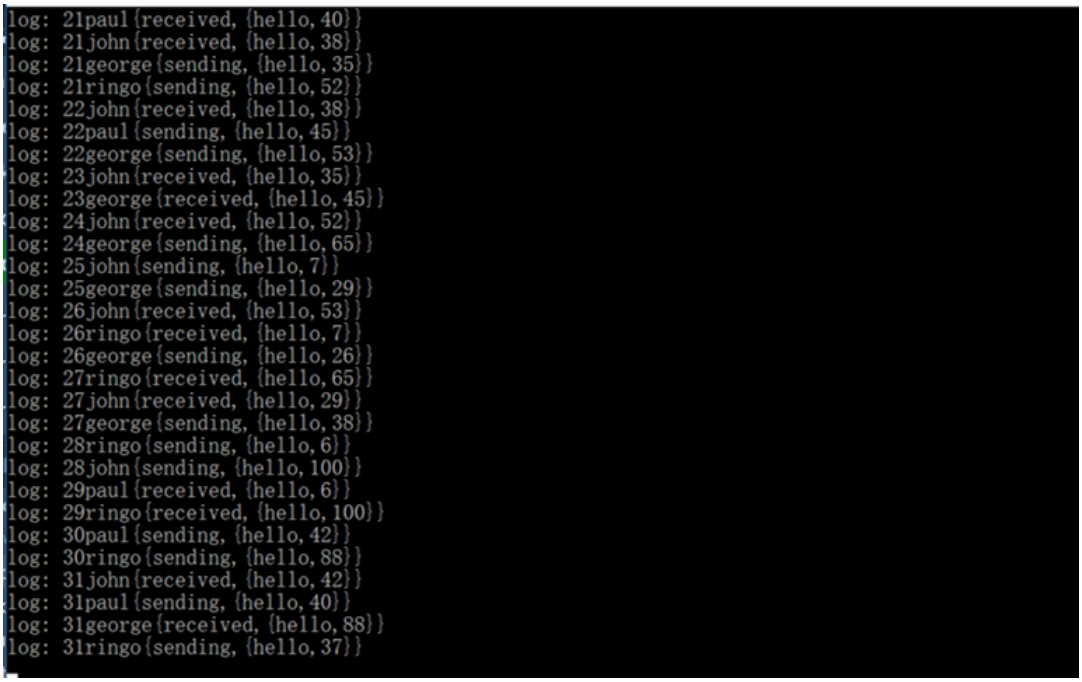

Figure 1: Output by Lamport algorithm

Also, the sending are strictly before receiving and the causal order is promised.

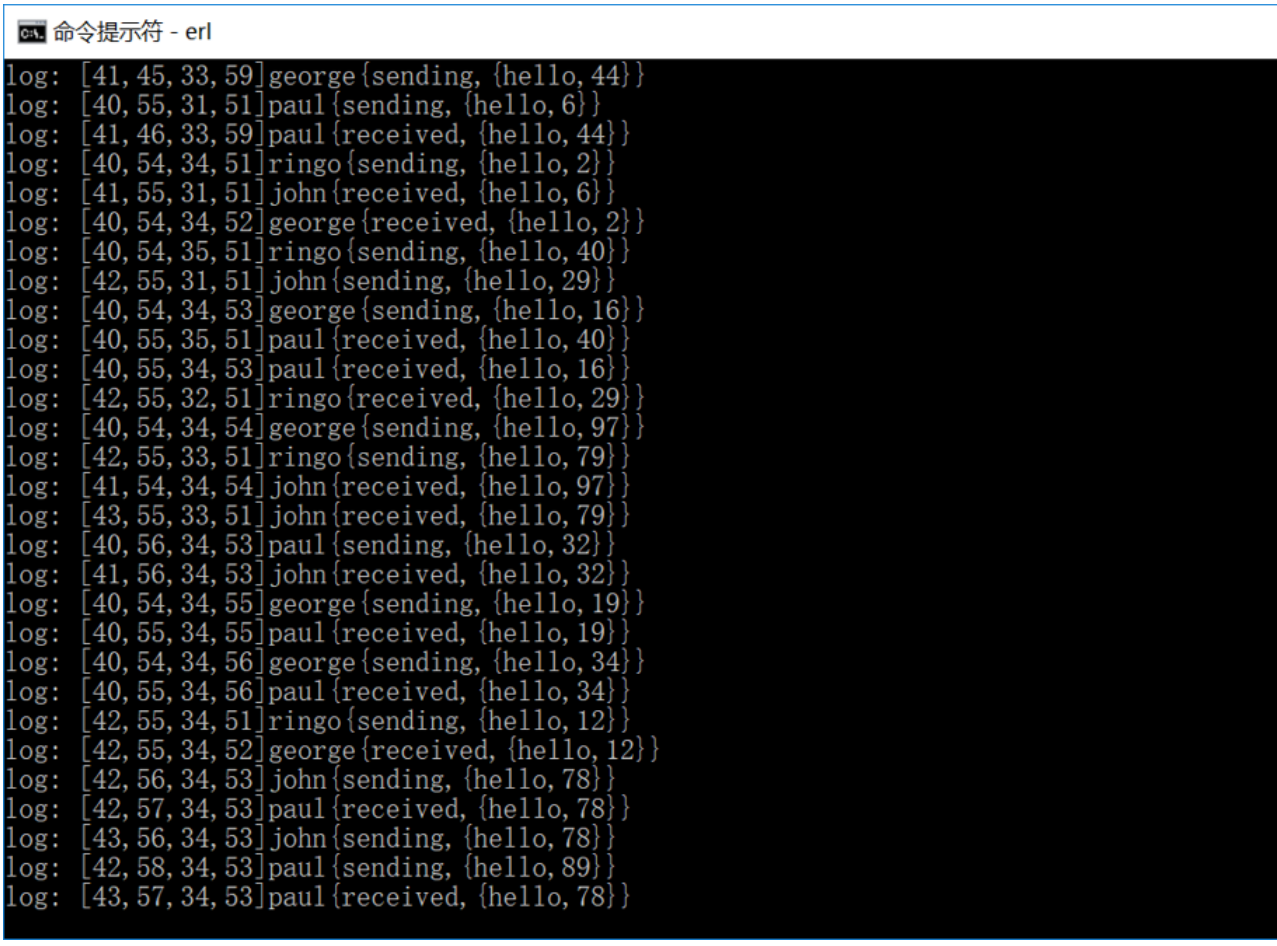

Figure 2: Output by Vector clock algorithm