HW3 Report, Loggy: A Logical Time Logger

David Fischer

September 20, 2025

1 Introduction

2 Main problems and solutions

2.1

2.2

2.3

3 Evaluation

3.1 Visualization

To aid evaluation, a module *mermaid* was implemented to create visual representations of the messages flowing between the vector clocks.

See appendix. (TODO, proper sendoff)

3.2 Delay caused by Jitter Comparison of Lamport and Vector

4 Conclusions

5 Appendix

Both of these tests were completed using test:run(¡module¿, 1500, 500).

5.1 Lamport Clock

```
119> test:run(time, 1500, 500).
loggy: starting with module time
log: s:5   ringo  sending  ( 24) c:1
log: s:5   john   sending  (  6) c:1
log: s:5   george sending  ( 26) c:1
log: s:2   paul   received ( 24) c:2
```

```
log: s:2   john   received ( 26) c:2
log: s:2   paul   received (  6) c:3
log: s:2   john   sending  ( 50) c:3
log: s:2   ringo  received ( 50) c:4
log: s:2   john   sending  ( 73) c:4
log: s:2   paul   sending  ( 28) c:4
log: s:8   george received ( 28) c:5
log: s:8   ringo  sending  (  2) c:5
log: s:8   john   sending  ( 37) c:5
log: s:13  george received ( 73) c:6
log: s:13  paul   received (  2) c:6
log: s:13  john   sending  (  1) c:6
log: s:3   george sending  ( 48) c:7
log: s:3   paul   sending  ( 30) c:7
log: s:3   ringo  received ( 48) c:8
log: s:3   paul   received ( 37) c:8
log: s:3   ringo  sending  ( 86) c:9
log: s:3   george received ( 86) c:10
log: s:3   george received ( 30) c:11
log: s:3   george sending  ( 85) c:12
log: s:3   ringo  received ( 85) c:13
log: s:3   ringo  sending  ( 83) c:14
log: s:3   george received ( 83) c:15
log: s:3   ringo  received (  1) c:15
```

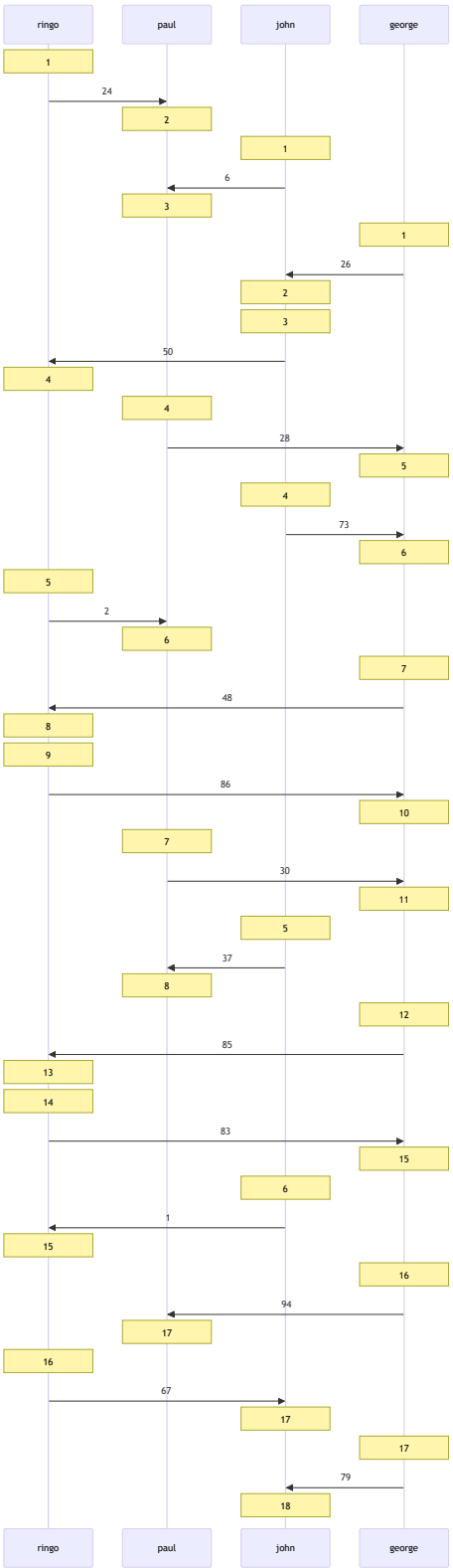

Figure 1: Sequence visualization of the Lamport timestamp algorithm

5.2 Vector Clock

```
120> test:run(vect, 1500, 500).
loggy: starting with module vect
log: s:1   ringo  sending  ( 24) c:ringo => 1
log: s:1   paul   received ( 24) c:paul => 1,ringo => 1
log: s:0   john   sending  (  6) c:john => 1
log: s:0   paul   received (  6) c:john => 1,paul => 2,ringo => 1
log: s:0   george sending  ( 26) c:george => 1
log: s:0   john   received ( 26) c:john => 2,george => 1
log: s:0   john   sending  ( 50) c:john => 3,george => 1
log: s:0   ringo  received ( 50) c:john => 3,ringo => 2,george => 1
log: s:2   john   sending  ( 73) c:john => 4,george => 1
log: s:0   paul   sending  ( 28) c:john => 1,paul => 3,ringo => 1
log: s:0   george received ( 28) c:john => 1,paul => 3,ringo => 1,george => 2
log: s:0   george received ( 73) c:john => 4,paul => 3,ringo => 1,george => 3
log: s:0   ringo  sending  (  2) c:john => 3,ringo => 3,george => 1
log: s:0   paul   received (  2) c:john => 3,paul => 4,ringo => 3,george => 1
log: s:0   george sending  ( 48) c:john => 4,paul => 3,ringo => 1,george => 4
log: s:0   ringo  received ( 48) c:john => 4,paul => 3,ringo => 4,george => 4
log: s:1   john   sending  ( 37) c:john => 5,george => 1
log: s:1   paul   received ( 37) c:john => 5,paul => 5,ringo => 3,george => 1
log: s:0   ringo  sending  ( 86) c:john => 4,paul => 3,ringo => 5,george => 4
log: s:0   george received ( 86) c:john => 4,paul => 3,ringo => 5,george => 5
log: s:2   george sending  (  8) c:john => 4,paul => 3,ringo => 5,george => 6
log: s:1   ringo  sending  ( 84) c:john => 4,paul => 3,ringo => 6,george => 4
log: s:1   paul   received ( 84) c:john => 5,paul => 6,ringo => 6,george => 4
log: s:1   paul   received (  8) c:john => 5,paul => 7,ringo => 6,george => 6
log: s:1   paul   sending  ( 46) c:john => 5,paul => 8,ringo => 6,george => 6
log: s:1   george received ( 46) c:john => 5,paul => 8,ringo => 6,george => 7
log: s:1   john   sending  (  1) c:john => 6,george => 1
log: s:1   ringo  received (  1) c:john => 6,paul => 3,ringo => 7,george => 4
log: s:0   george sending  ( 99) c:john => 5,paul => 8,ringo => 6,george => 8
log: s:0   paul   received ( 99) c:john => 5,paul => 9,ringo => 6,george => 8
```

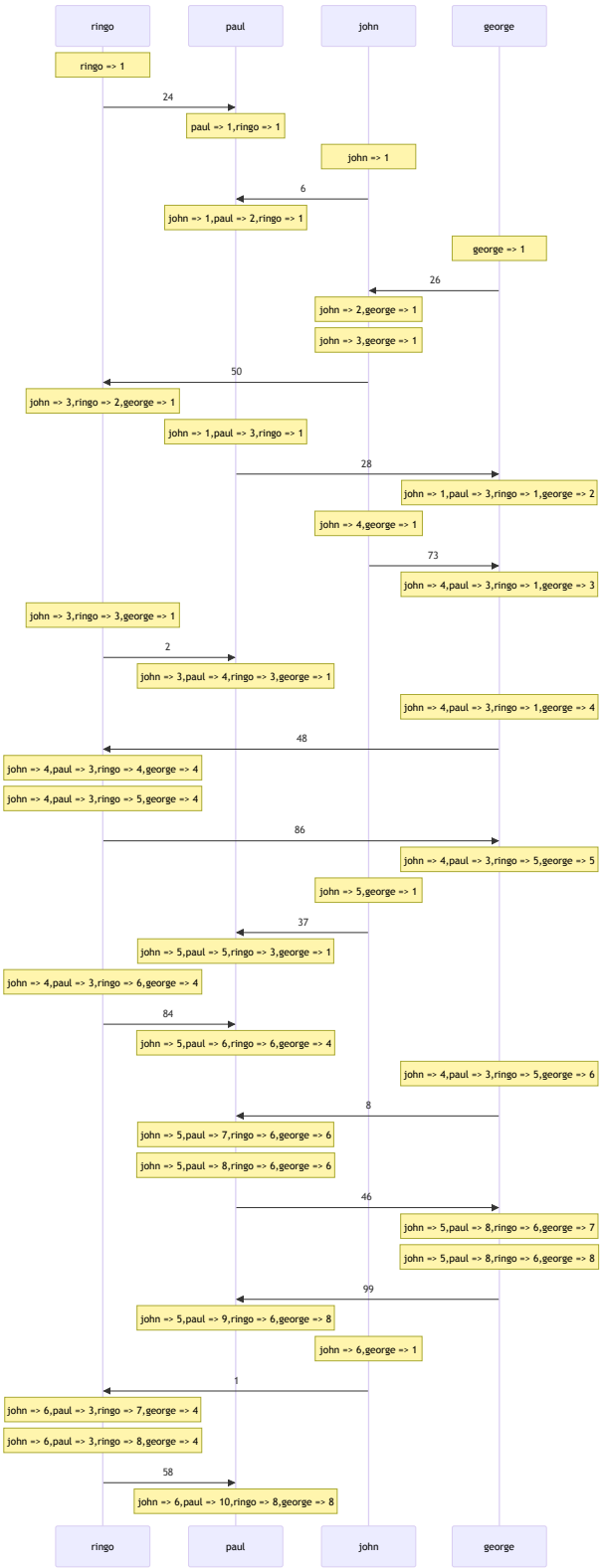

Figure 2: Sequence visualization of the vector clock implementation