Implementation Of Lamport's Scalar Clocks

Surekha Busa
Department of computer science
Kent State University
Kent,Ohio
sbusa@cs.kent.edu

Abstract—**A logical clock is a mechanism for capturing chronological and causal relationships in a distributed system. The paper discusses about the implementation of logical clocks by lamport's. The paper majorly focuses on clock updates and analyze its dependency on the number of processes and messages.**

I. INTRODUCTION

A distributed system consists of a collection of processes that communicate by passing messages to each other. So, each *event* in a distributed system is either a *local step* of a process, a *send* event, or a *receive* event. In studying a distributed system, we often need to assign an order to these events. Therefore, we use the concept of clocks for synchronization and ordering of events. Physical clocks capture the event ordering but suffer from pitfalls like clock drift so we use logical clocks. This paper deals with the logical clock implemented by lamport's.

II. LAMPORT'S SCALAR CLOCKS

Each process P_i has a logical clock C_i assigned with an integer, the assigned value is the *timestamp* of this event, denoted as $C(a)$. The timestamp increases each time an event occurs. Consider two event a and b , if event a occurs before b i.e. *if a -> b, then C(a) < C(b)* this is called the consistency. Strongly consistency occur when *C(a) < C(b) then a -> b* .The major problem with scalar locks is they cannot order concurrent events i.e. it is *partially consistent*.

A. Implementation Rule

- Everytime a process send a message, it updates its clock Ci := Ci + d (d>0) and tag this clock to the outgoing message.
- Upon receive of message, increments its clock and compares its clock With the clock in the message myclock = MAX(message, myclock).

III. EXPERIMENT

A. Base Algorithm

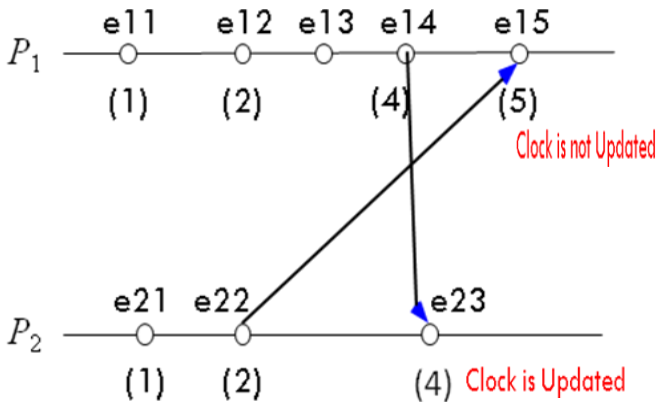
Base algorithm: *Random Flood*

- There are N processes in the network, The number is taken as an input from the user
- There is one initiator which sends messages to random nodes.
- Upon receipt of a message, the process randomly chooses a process and sends a message.

This base algorithm terminates when every process send a message.

B. Experimental SetUp

The experiment is done considering the clock updates Whenever, the clock of the receiver takes the value of the clock of the message we say *the clock is updated*.



The clock updates depend on 1: The number of messages in the computation. 2: The number of processes in the system.

Considering this, the experiment was set up varying the number of processes ranging from 10-100.The clock updates

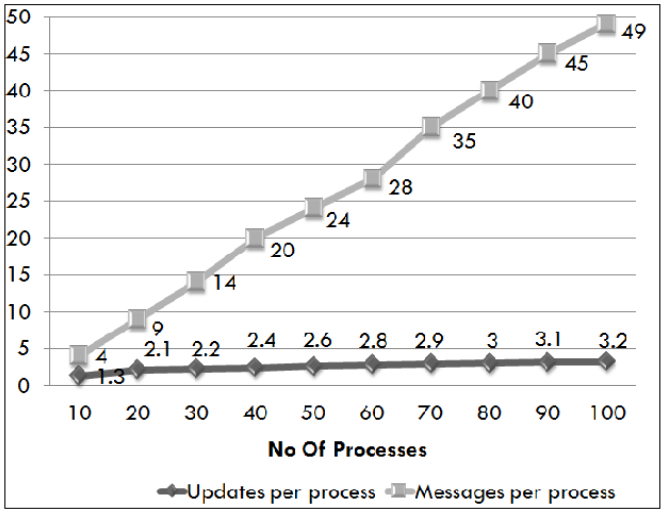
and number of messages was recorded for each 10 runs and average values are taken. Similar experiment is conducted by calculating the total updates per process (total updates/no. of processes) and comparing it by varying the number of processes

No. Of Processes	No. Of Updates	No. Of Messages	Updates Per Process	Messages Per Process
10	13	46	1.3	4
20	41	197	2.05	9
30	65	442	2.17	14
40	95	829	2.37	20
50	131	1230	2.62	24
60	169	1724	2.81	28
70	206	2509	2.94	35
80	240	3238	3	40
90	278	4081	3.08	45
100	319	4938	3.19	49

IV. Result Analysis And Observations

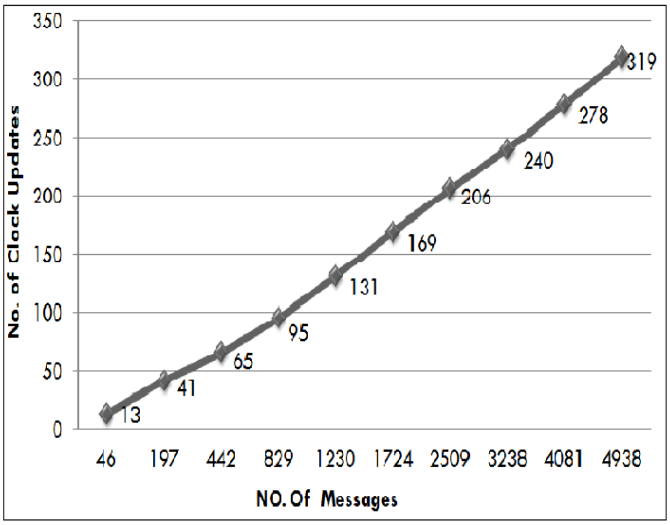
A. No. Of Updates and Messages Varying No. Of Processes.

The below graph depicts the total no. of messages and clock updates generated by varying the number of processes.we observe a increases in the updates and messages as the no. of processes increase.



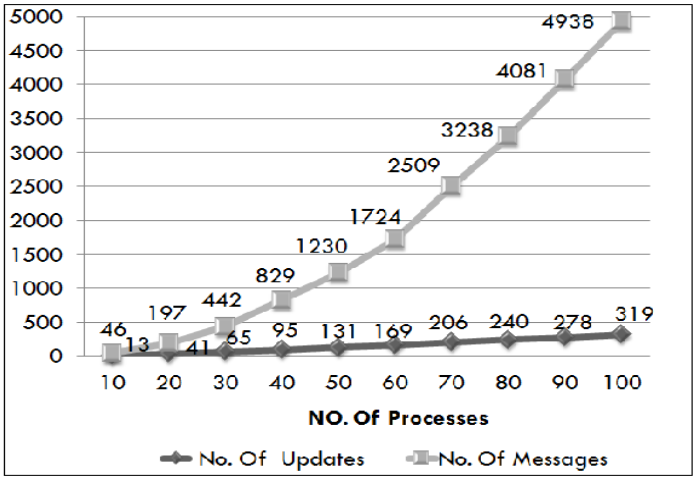
B. No. Of Updates and Messages Per Process Varying No. Of Processes

The below graph depicts the no. of updates and messages per process by varying the number of processes. we observe a increase in the updates and messages as the no. of processes increase.



C. *No. Of Messages Versus No. of Clock Updates*

The below graph depicts the no. of updates by varying the number of messages. we observe a increase in the updates as the no. of messages increase.



V. Conclusion and Future Work

From the data results obtained we conclude that the clock updates increases with the increase in total number of processes and messages.

For my future work I would like to check the performance of logical clocks with large no. of processes and also in various topologies.

ACKNOWLEDGEMENT

I would like to Thank Dr. Nesterenko for his support throughout the course and my friends for their patience

REFERENCES

[1] Leslie Lamport (1978). ”Time, clocks, and the ordering of events in a distributed system”. Communications of the ACM 2 (7): 558-565.

[2] Mukesh Singhal,Niranjan G. Shivaratri “Advanced Topics in Operating Systems”. McGRAW-Hill international edition: Chapter 5.