DISTRIBUTED SCHEDULING PROJECT

Tushita Sharva
CS21BTECH11022

February 29, 2024

1 Introduction

In this report, we present an analysis of the performance of Vector Clocks and the Singhal-Kshemkalyani Optimization implemented in a distributed system based on programs we implemented. The goal of our study is to compare the overheads incurred with message storage and exchange in both algorithms.

2 Implementation Details

2.1 Environment Setup

- Programming Language: C++
- Framework used: MPI
- Operating Environment: Local machine

3 Experimental Setup

- We varied the number of processes (n) from 10 to 15 in increments of 1 while keeping other parameters constant.
- We executed both algorithms multiple times to gather statistically significant results.
- Each execution used the same graph topology with randomly assigned neighbors.

4 Results and Analysis

4.1 Space Utilization

We measured the space utilized by each process for storing the vector clocks and sending messages for both algorithms. The results showed that Singhal-Kshemkalyani Optimization achieved lower space utilization compared to the traditional Vector Clocks due to its optimized message structure.

4.2 Average Number of Entries in Each Message

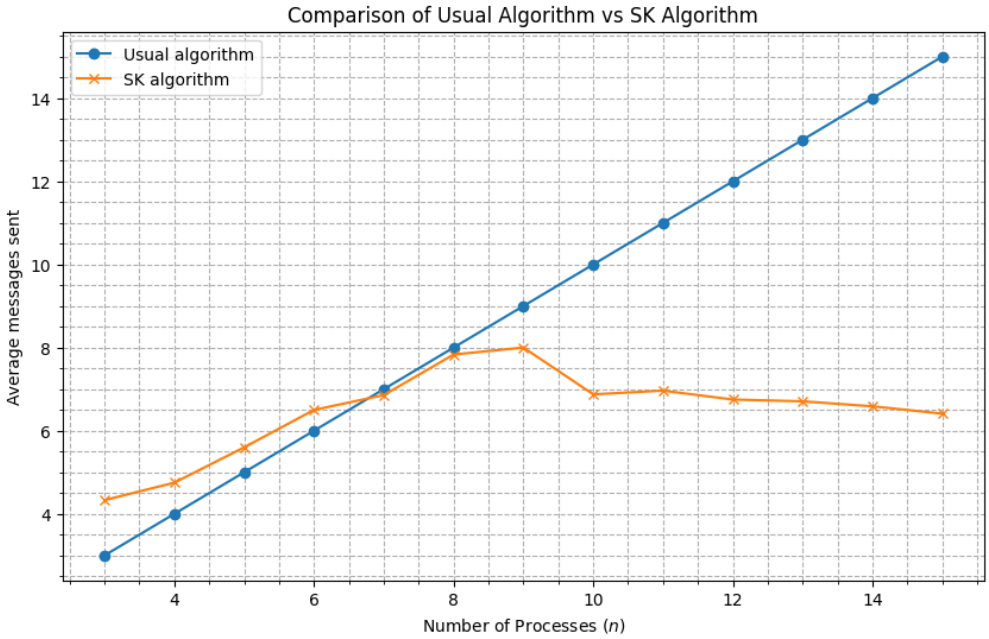
The below table gives the average number of entries in each message sent by a process. I calculated it in the following way:

- First, each process would take note of how many entries are there in the message sent by it.
- Each process would divide the value of number of entries with number of messages it sent to get average number of entries per message.
- They will be printed to `space-SK.log` and `space-VC.log` files. We will add each of them and divide with number of processes to get average across all the processes.

Number of processes (n)	Usual algorithm	SK algorithm
3	3	4.33
4	4	4.75
5	5	5.6
6	6	6.5
7	7	6.857
8	8	7.833
9	9	8
10	10	6.875
11	11	6.964
12	12	6.750
13	13	6.707
14	14	6.587
15	15	6.411

4.3 Graphical Representation

Below is a plotted graph with the number of processes (n) on the x-axis and the average number of entries in each message on the y-axis. The graph clearly illustrates the decreasing trend of the average number of entries for Singhal-Kshemkalyani Optimization compared to Vector Clocks as the number of processes increases.



5 Conclusion

This analysis demonstrates the effectiveness of Singhal-Kshemkalyani Optimization in reducing message overhead compared to traditional Vector Clocks.

- We can see that in the beginning SK algorithm seems ineffective. The reason is that for smaller numbers, for example 3, the messages sent in **VC algorithm** is constantly **3**, but in **SK algorithm** it can vary from **0 to 6**. Hence the overhead incurred.
- But as **number of processes increases**, we can see that SK has optimized message complexity whereas there is a linear increase in the size of messages sent in the original VC algorithm.