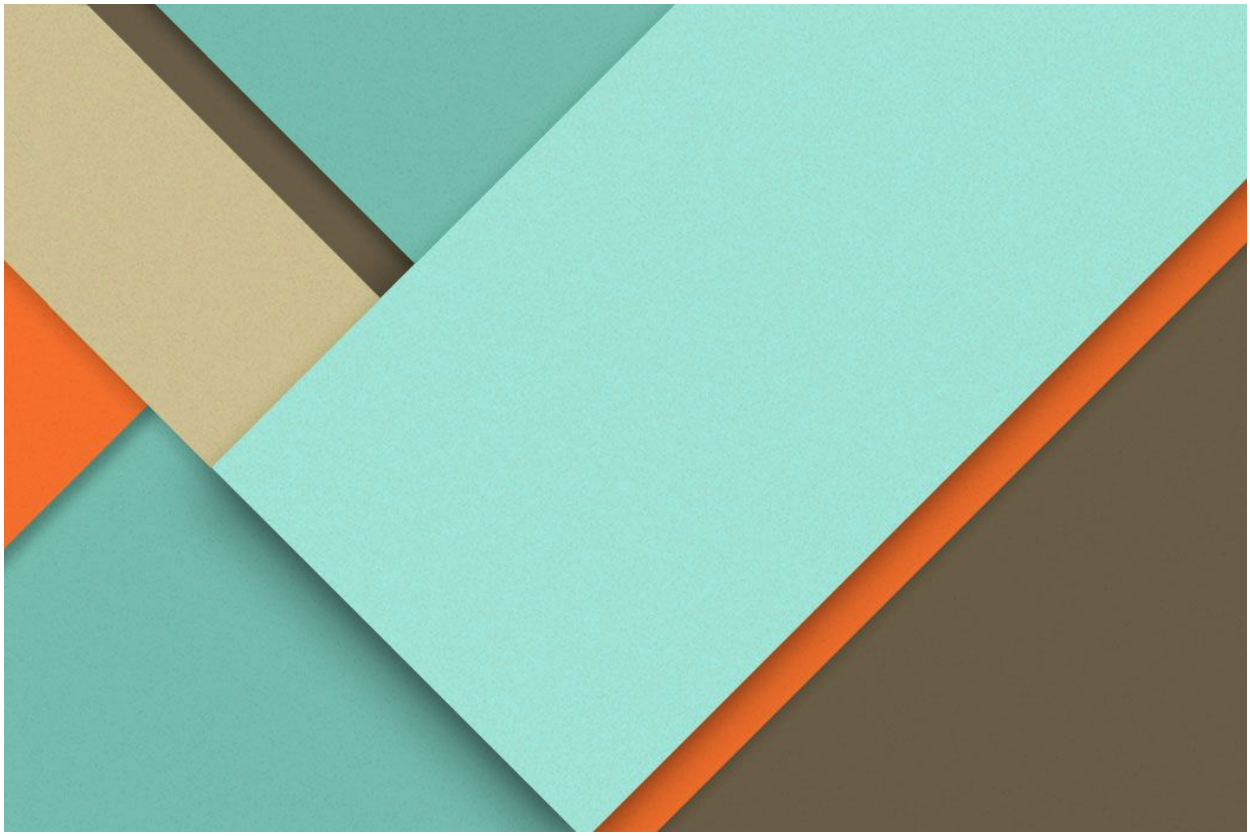


ASSIGNMENT-1

25.08.2018

OM SITAPARA
CS16BTECH11036

Overview

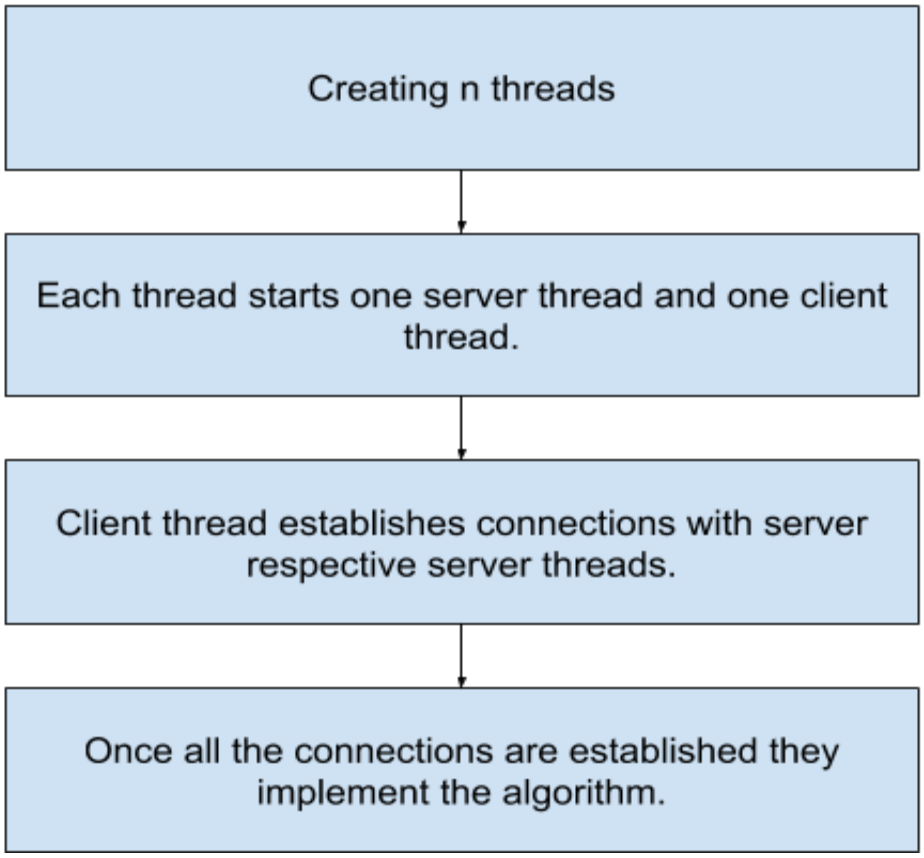
The goal of this assignment was to implement vector-clocks and Singhal-Kshemkalyani optimization on a Distributed System in C++.

Goals

1. Compare the overheads incurred with message stored and exchanged.
2. To output space utilized by each process for storing the vector clocks and sending messages by both the algorithms.

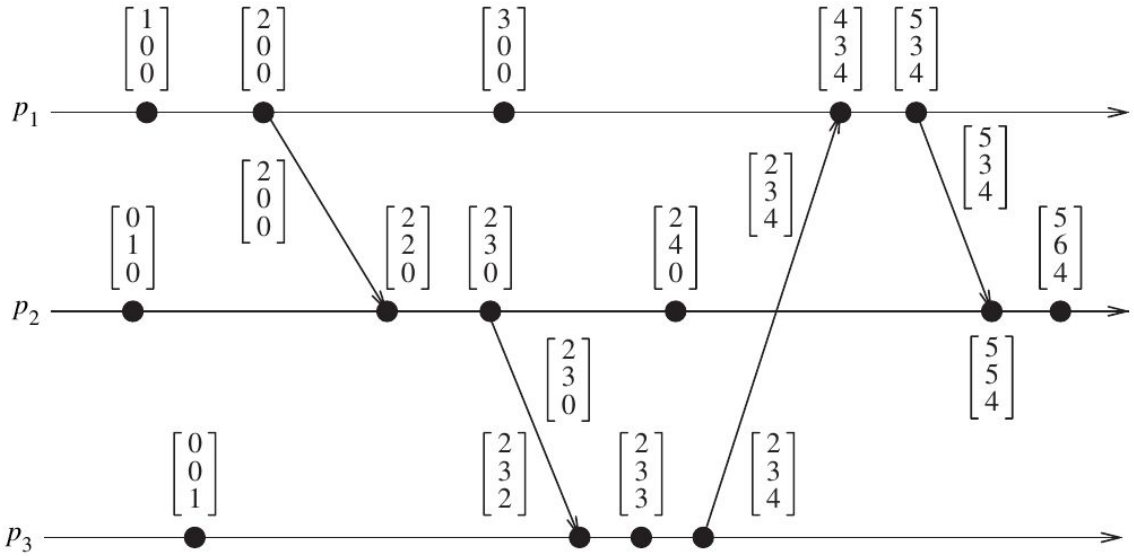
Approach

For creating a distributed system i have used socket programming for making connections between multiple client and server. Flow chart of connections:



Algorithm 1 : Normal Vector clock

In the normal vector clock when one process has to send message to another process it sends its whole vector clock to that process and the receiver process updates its vector clock according to this condition : max(vt[i], vt_j[i]). An example can be seen here :



On the occurrence of an internal event it first increments its clock and than sends the message also when it receives the message it increments its vector clock.

Algorithm 2 : Singhal-Kshemkalyani optimization

In this algorithm unlike the normal vector clock algorithm only those values of vector clocks are send which were changed since the last message. For its implementation each process maintains two more vectors namely $LS_i[1....n]$ **Last Send** and $LU_i[1...n]$ **Last Update.** This vectors stores:

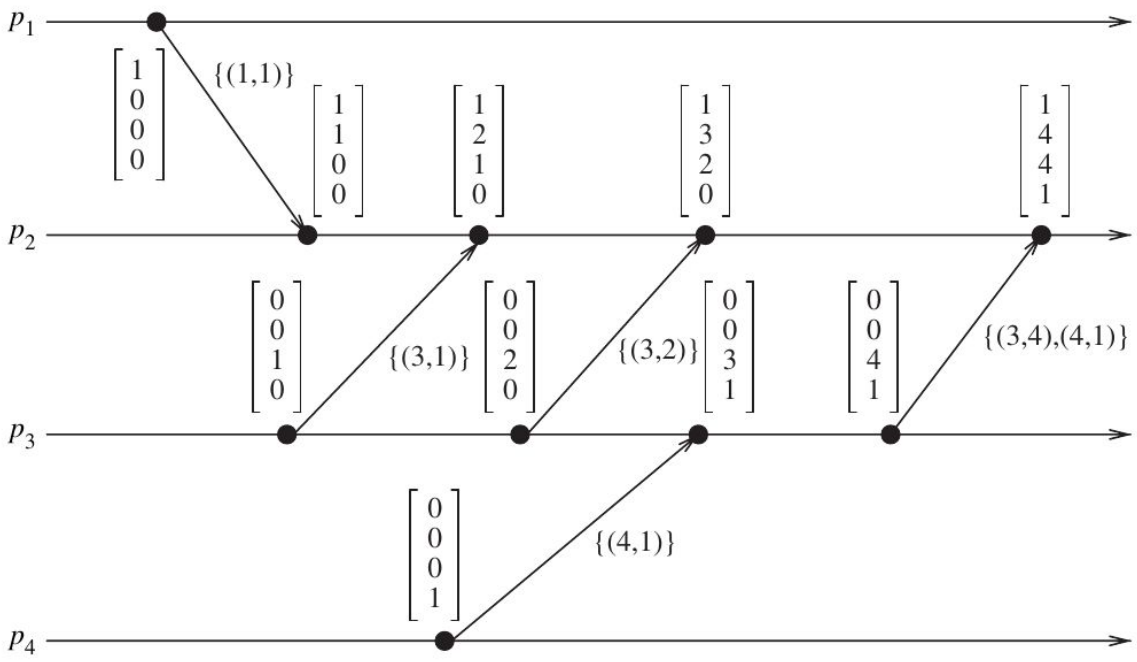
$LS_i[j]$ indicates the value of $vt_i[i]$ when process p_i last sent a message to process p_j .

$LU_i[j]$ indicates the value of $vt_i[i]$ when process p_i last updated the entry $vt_i[j]$.

So it sends message in tuples according to this condition :

$$\{(x, vt_i[x]) \mid LS_i[j] < LU_i[x]\}$$

An example can be seen here :



Properties

- Both the vector clocks are strongly consistent.
- So both of them follows following conditions :

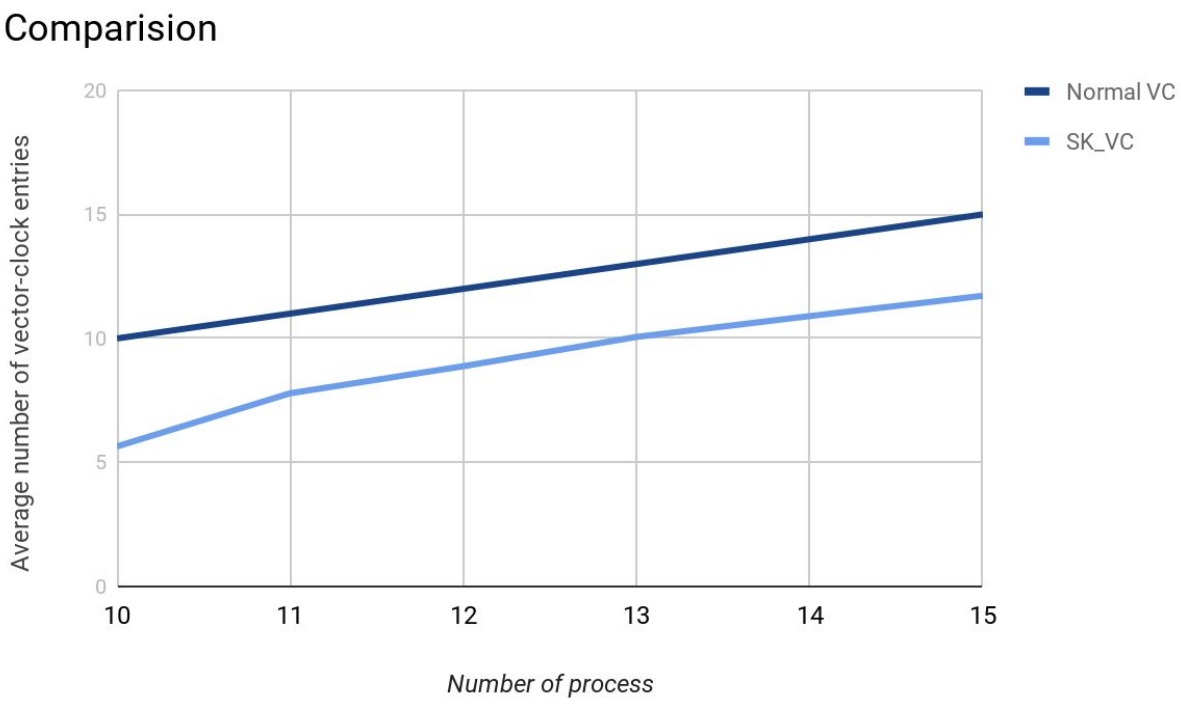
  - **x → y ⇔ vh < vk**
  - **x y ⇔ vh vk**

Observations and Conclusions

- The space utilized by normal vector clock to store logical time : **O(n)** where **n** is number of process.
- The space utilized by the Singhal-Kshemkalyani optimization to store logical time is : **O(3n)** where **n** is number of process.

Here we plot the graph for number of process as x-axis and average number of message passed be y-axis.

Graph:



- Here from the graph we can see that, average number of entries in vector clock linearly increase with n : **Vector clock entries ∝ n.**
- The number of entries in optimisation technique is proportional to n : **SK_VC ∝ n.**
- The number of entries in **SK_VC < Normal_VC.**

Challenges Faced

- To implement networking and sockets for multiple server and clients for first time.
- If a process server is not active yet and a receiver process tries to connect than connection fails so to handle this my client waits until all the servers are active.
- Vector clock updation needs to been done inside locks as there might be case when a process can have internal event and a message receive event.
- The exit condition for the program when all the process have sent *m* messages or not and closing the sockets for client first and than exiting the server was achieved by using atomic variable.