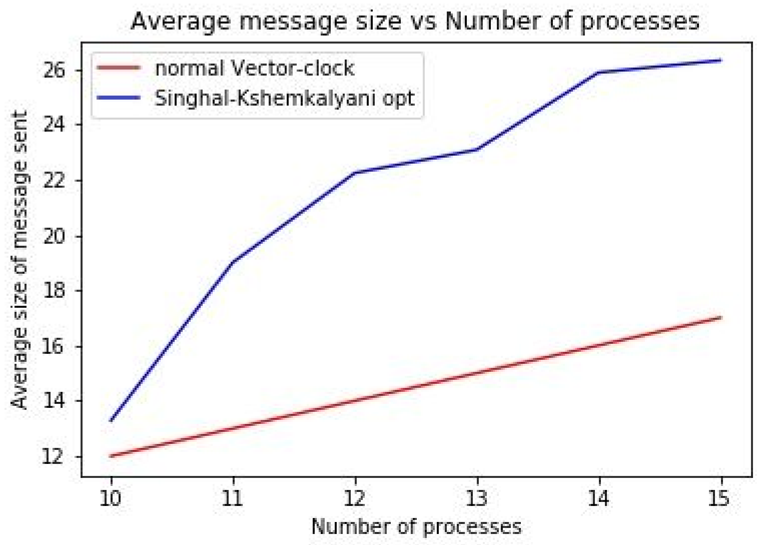
Report

Design

- Threads were used to simulate different processes
- 2 threads were created for each process, one simulating internal events and sending messages to other processes, and the other listening for others messages
- TCP was used to communicate between the processes. A separate socket was created for each process and a different port was used to bind it with for different processes.
- The message number and the sender process is piggybacked with every message sent to other processes, this is done for both the naive vector clock aapproach and singhal kshemkalyanis optimization and hence should not affect the analysis of the optimization.

Graphs and Analysis

Worst Case topology was chosen



The graph topology chosen was fully connected, and the m messages were sent in a round robin manner to the neighbours of each process.

Singhal Kshemkalyanis optimization works well when there are relatively few updates made to the vector clock of the process since the last time it sent a message to another process.
This holds true when we have sparsely connected graphs, and the communication is localized only to few of its neighbours.

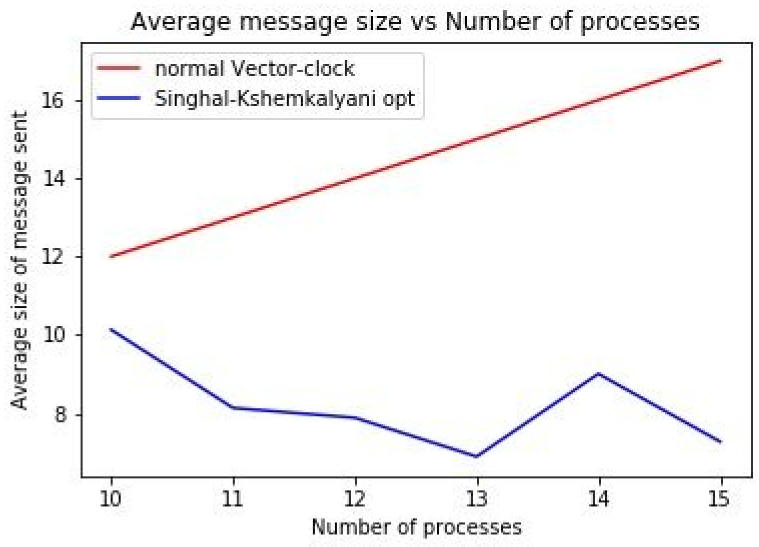
We can see in the worst case singhal kshemkalyanis optimization have about twice the bandwidth requirement compared to vanilla vector clock approach.

Reason:

This is because in the worst case all the vector entries get updated since last sent message to a process, and twice the number of entries need to be sent in the optimization since it sends tuple of index(process id of changed vector clock entry) and its value compared to the vanilla vector clock approach which sends n entries no matter the graph topology.

It can be clearly seen from the logs, the consistency property is satisfied by both the programs.

Better Case:



A sparse graph was used as the topology to get the above graph.

It can be clearly seen that as the number of processes increases and are connected to only a few nodes as compared to the size of the graph, the optimization is highly useful for cutting down the bandwidth requirement.

It can also be observed that for less number of processes, and a sparse graph singhal kshemkalyanis technique perform almost as well(or worse) as the naive method.

Reason:

As the graph is sparsely connected only a few vector entries change since the last time a message is sent to a process’s neighbour, hence the optimization technique becomes beneficial.

Errors encountered

Errors Encountered

EAGAIN : Timeout in accept function
EADDRINUSE : when the program is run multiple times, we need to give some time in between the runs
ECONNREFUSED : error in connection function, the socket refuses the connection.