Lamport’s Logical Clocks

Project 1

CPSC 474

19 September 2020

Dr. Doina Bein

David Nguyen 891362089

**Summary**

For this assignment, I use multithreading for calculating and verifying the Lamport Clock.

For the calculation, a single thread each event will be given a count *k* until it reaches a “dead-end” that will halt their execution (e.g. receive event with no send event). For a M x N matrix (internal, receive, send) events there will be N threads that will run in parallel. All threads will calculate each event until there is no more left. If one or more threads reaches a dead-end then the dead-end threads will have to wait for the corresponding send event and the send-event thread to finish executing. Then the dead-end threads will calculate each event again until it finishes or if there are more dead-end threads, then the cycle will repeat itself again.

**Pseudocode**

//M x N matrix of internal, send, receive event

**Calculate**

//spawn N threads

vector<thread> threads;

for( i to N){

for( I to M){

}

}

//If *a* is the first event and is an internal or send event, then LC(*a*) = 1

if(events)

. //If *a* is the first event and is a receive event, then LC(*a*) = *k* + 1 where *k* is the LC-value of the send event corresponding to *a* (that has occurred at a process other than P).

else if()

//If *a* is not the first event and is an internal or send event, then LC(*a*) = *k* + 1 where *k* is the LC-value of the event just before *a* at process P.

else if()

//If *a* is not the first event and is a receive event, let *b* be the send event corresponding to *a* (that has occurred at a process other than P) and *k* be the clock value of the event just before *a* at process P. Then LC(a) = max{ k, LC(b) } + 1

else if()  
  
 return;

}

**Verify**

**Screenshots**